Comparison of Point Sets and Sequences for Quasi-Monte Carlo and for Random Number Generation *

Pierre L'Ecuyer[1]

DIRO, CIRRELT, and GERAD,
Université de Montréal, Canada
http://www.iro.umontreal.ca/~lecuyer

Abstract. Algorithmic random number generators require recurring sequences with very long periods and good multivariate uniformity properties. Point sets and sequences for quasi-Monte Carlo numerical integration need similar multivariate uniformity properties as well. It then comes as no surprise that both types of applications share common (or similar) construction methods. However, there are some differences in both the measures of uniformity and the construction methods used in practice. We briefly survey these methods and explain some of the reasons for the differences.

1 Introduction

1.1 Random Number Generators

(Pseudo)Random number generators (RNGs) are typically defined by a (deterministic) recurring sequence in a finite state space $\mathcal{S}$, usually a finite field or ring, and an output function mapping each state to an output value in $\mathcal{U}$, which is often either a real number in the interval $(0, 1)$ or an integer in some finite range [1–4]. We shall assume here that each output is a real number in $(0, 1)$ and that the purpose of the RNG is to mimic a sequence of independent $U(0, 1)$ random variables (i.e., uniformly distributed over $(0, 1)$). Of course, with such an algorithmic construction, this can only be a fake. The quality of the fake should be measured in a way that depends on the application.

One could argue that physical noise would provide a safer source of (true) randomness. With careful design, it does, and it is appropriate for certain applications such as generating random keys in cryptology and for gaming machines, for example, where unpredictability is a major requirement. In this paper, we focus on RNGs for simulation applications, where fast algorithmic RNGs are much more convenient than physical sources, because they are faster, they permit one to replicate the same exact sequence several times (this is often needed in

* This work was supported NSERC-Canada Grant Number ODGP0110050 and by a Canada Research Chair to the author.

simulation, for example for comparing similar systems and in optimization and variance reduction algorithms [5–7]), and they do not require special hardware.

Let $s_0, s_1, s_2, \ldots$ denote the successive states of our RNG, and $u_0, u_1, u_2, \ldots$ be the corresponding sequence of output values. After selecting s_0 (which might be random), the successive states follow the deterministic recurrence $s_i = f(s_{i-1})$ for a given *transition function* $f : \mathcal{S} \to \mathcal{S}$, and the *output* at step i is $u_i = g(s_i)$ for some *output function* $g : \mathcal{S} \to (0, 1)$. This output sequence is necessarily (eventually) periodic, with period ρ that cannot exceed the number of distinct states, $|\mathcal{S}|$. When the state occupies b bits of memory, we have $\rho \leq 2^b$ and we usually require that ρ be close to 2^b, to avoid wasting memory. Values of b for certain practical RNGs can be as much as 20,000 or even more [8–10], but for simulation, a few hundred is probably large enough.

Besides a long period, other standard requirements for RNGs include high running speed (in 2008, fast RNGs can produce over 100 million $U(0, 1)$ random numbers per second on a standard computer), reproducibility and portability (the ability to reproduce the same sequence several times on the same computer, and also on any standard computing platform), and the possibility to quickly jump ahead by an arbitrarily large number of steps in the sequence. The latter is frequently used to split the sequence into several shorter (but still very long) disjoint subsequences, in order to provide an arbitrary number of virtual generators (often called RNG streams) [5, 6].

However, these basic properties are not sufficient To see why, just note that an RNG defined by $u_i = s_i = (i+1/2)/2^{10000} \bmod 1$ easily satisfies all the above properties, and the values cover the interval $(0, 1)$ very evenly in the long run, but this RNG certainly fails to provide a good imitation of independent $U(0, 1)$ random variables. The problem is the lack of (apparent) *independence* between successive values.

A key observation is that we have both uniformity and independence if and only if for any integer $s > 0$, $(u_0, \ldots, u_{s-1})$ is a random vector with the uniform distribution over the s-dimensional unit hypercube $(0, 1)^s$. We want the RNG to provide a good approximation of this property. If the seed s_0 is selected at random in $\mathcal{S}$, then the vector $(u_0, \ldots, u_{s-1})$ has the uniform distribution over the finite set $\Psi_s = \{(u_0, \ldots, u_{s-1}) : s_0 \in \mathcal{S}\}$, which can be viewed as a discrete approximation of the uniform distribution over $(0, 1)^s$. For this approximation to be good, Ψ_s must provide a dense and uniform coverage of the unit hypercube $(0, 1)^s$, at least for moderate values of s. This is possible only if $\mathcal{S}$ has large cardinality. In fact, this is a more important reason for having a long period than the risk of exhausting the RNG cycle. More generally, we want high uniformity of the sets of the form $\Psi(\mathfrak{u}) = \{(u_{i_1}, \ldots, u_{i_d}) : s_0 \in \mathcal{S}\}$, where $\mathfrak{u} = \{i_1, \ldots, i_d\}$ is an arbitrary set of integers such that $0 \leq i_1 < \cdots < i_d$.

But how should we measure the uniformity of Ψ_s (and of the sets $\Psi(\mathfrak{u})$)? There are many ways. But a crucial requirement is that the selected measure of uniformity must be computable efficiently without generating the random numbers (or enumerating the points of Ψ_s) explicitly, because there are just too many of them. For nonlinear RNGs, easily computable measures of uniformity

are difficult to find. This is the main reason why most good RNGs used in practice are based on linear recurrences [3, 9]. Specific measures for linear RNGs are discussed later in this paper. The choice generally depends on the type of RNG, for computability reasons.

To construct a new RNG, one would usually specify a parameterized class of (large-period) constructions (based on the availability of a fast implementation) and a figure of merit, and then perform a computer search for the construction with the largest figure of merit that can be found within that class. Then, the selected RNG is implemented and tested empirically. There is a large variety of *empirical statistical tests* for testing the statistical behavior of RNGs [1, 9].

1.2 Low-discrepancy Sets and Sequences

In *quasi-Monte Carlo* (QMC) numerical integration, we want a set of n points, $P_n = \{\mathbf{u}_0, \ldots, \mathbf{u}_{n-1}\}$, that cover the s-dimensional unit hypercube $[0,1)^s$ very evenly. These points are used to approximate the integral of some function $f : [0,1)^s \to \mathbb{R}$, say

$$\mu = \int_{[0,1)^s} f(\mathbf{u}) d\mathbf{u},$$

by the average

$$\bar{\mu}_n = \frac{1}{n} \sum_{i=0}^{n-1} f(\mathbf{u}_i). \tag{1}$$

This μ can be interpreted as the mathematical expectation $\mu = \mathbb{E}[f(\mathbf{U})]$, where $\mathbf{U}$ is a random vector uniformly distributed over $[0,1)^s$. Here, the cardinality n of the point set is much smaller than for RNGs, because we need to evaluate f at each of these points. It rarely exceeds a million.

Again, a key question is: How do we measure the uniformity of the point set P_n? Standard practice is to use a figure of merit that measures the *discrepancy* between the empirical distribution of P_n and the uniform distribution over $[0,1)^s$ [11–14, 4]. Many of these discrepancies are actually defined as the worst-case integration error, with P_n, over all functions f whose *variation* $V(f) = \|f - \mu\|$ does not exceed 1, in a given Banach (or Hilbert) space $\mathcal{H}$ of functions [11, 15]. In this setting, the worst-case error can be bounded by the product of the discrepancy $D(P_n)$, that depends only on P_n, and the function's variation, that depends only on f:

$$|\bar{\mu}_n - \mu| \le D(P_n) V(f) \tag{2}$$

for all $f \in \mathcal{H}$. This is a generalized form of the Koksma-Hlawka inequality [4]. Generally speaking, for a given point set, the more restricted (or smoother) the class of functions f for which $V(f) \le 1$, the smaller will be the discrepancy, and the faster $\min_{P_n} D(P_n)$ will converge to 0 as a function of n. That is, the worst-case error bound for the best possible point set will converge to 0 more quickly. Specific examples of discrepancies are mentioned in Section 2.

Here, because n is not so large, a computing time of $O(n)$ or even $O(n^2)$ for the discrepancy is acceptable (in contrast to RNGs). The choice of discrepancy can be different for this reason.

In practice, the worst-case error bound (2) is usually much too difficult to compute or even to approximate, and may be very loose for our specific function f. For these reasons, one would rather turn the deterministic approximation $\bar{\mu}_n$ into a randomized unbiased estimator, and replace the error bound by a probabilistic error estimate obtained by estimating the variance of the estimator. This is achieved by randomizing the point set P_n so that [16, 17, 13, 18]:

(a) it retains its high uniformity when taken as a set and
(b) each individual point is a random vector with the uniform distribution over $(0,1)^s$.

A *randomized QMC* (RQMC) estimator of μ, denoted $\hat{\mu}_{n,\mathrm{rqmc}}$, is defined as the average (1) in which the $\mathbf{u}_i$ are replaced by the n randomized points. By performing m independent replicates of this randomization, and computing the sample mean and sample variance of the m independent realizations of $\hat{\mu}_{n,\mathrm{rqmc}}$, one obtains an unbiased estimator of μ and an unbiased estimator of the variance of this estimator. This permit one to obtain an asymptotically valid confidence interval on μ [17, 13].

A simple randomization that satisfies these conditions is a *random shift modulo 1* [19, 17, 20]: Generate a single point $\mathbf{U}$ uniformly over $(0,1)^s$ and add it to each point of P_n, modulo 1, coordinate-wise. Another one is a *random digital shift in base b* [21, 13, 22]: generate $\mathbf{U}$ uniformly over $(0,1)^s$, expand each of its coordinates in base b, and add the digits, modulo b, to the corresponding digits of each point of P_n. For $b = 2$, this amounts to applying a coordinate-wise exclusive-or (xor) of $\mathbf{U}$ to all the points.

The variance of the RQMC estimator is the same as its mean square error, because it is unbiased, so by squaring on each side of (2) we obtain that

$$\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}] \le V^2(f) \cdot \mathbb{E}[D^2(P_n)],$$

so the variance converges at least as fast as the mean square discrepancy.

A *low-discrepancy sequence* is an infinite sequence $\{\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2, \ldots\}$ such that the point set P_n formed by the first n points of the sequence has low discrepancy for all large enough n. Often, the discrepancy is lower along a subsequence of values of n; e.g., if n is a power of a given integer. Such sequences are convenient if we want to increase n (adding more points to the set) until some error bound or error estimate is deemed small enough, instead of fixing n a priori. A low discrepancy point set of sequence can also be *infinite-dimensional*; i.e., each point has an infinite sequence of coordinates. In this case, the sequence of coordinates are usually defined by a recurrence. This is convenient when $f(\mathbf{U})$ depends on a random and unbounded number of uniform random variables, which is frequent.

1.3 Importance of low-dimensional projections

When the dimension s is large, filling up the s-dimensional unit cube evenly requires too many points. For $s = 100$, for example, we already need 2^{100} points

just to have one point near each corner of the hypercube. For quasi-Monte Carlo, this is impractical. For RNGs, we can easily have more than 2^{100} points in Ψ_s, but the high-dimensional uniformity eventually breaks down as well for a larger s.

In QMC, we are saved by the fact that f is often well approximated by a sum of low-dimensional functions, that depend only on a small number of coordinates of $\mathbf{u}$; that is,

$$f(\mathbf{u}) \approx \sum_{\mathfrak{u} \subseteq \mathcal{J}} f_{\mathfrak{u}}(\mathbf{u}), \tag{3}$$

where each $f_{\mathfrak{u}} : (0,1)^s \to \mathbb{R}$ depends only on $\{u_j,\, j \in \mathfrak{u}\}$, and $\mathcal{J}$ is a family of small-cardinality subsets of $\{1, \ldots, s\}$. Then, to integrate f with small error, it suffices to integrate with small error the low-dimensional functions $f_{\mathfrak{u}}$ making up the approximation. For that, we only need high uniformity of the projections $P_n(\mathfrak{u})$ of P_n over the low-dimensional sets of coordinates $\mathfrak{u} \in \mathcal{J}$. This suggests a figure of merit defined as a weighted sum (or supremum) of discrepancy measures computed over the sets $P_n(\mathfrak{u})$ for $\mathfrak{u} \in \mathcal{J}$. The figures of merit used to select QMC point sets are typically of that form.

This heuristic interpretation can be made rigorous via a functional ANOVA decomposition of f [23, 18, 24]. When (3) holds for $\mathcal{J} = \{\mathfrak{u} : |\mathfrak{u}| \leq d\}$ for a small d, we say that f has low effective dimension in the superposition sense, while if it holds for $\mathcal{J} = \{\mathfrak{u} \subseteq \{1, \ldots, d\}\}$ for a small d, we say that f has low effective dimension in the *truncation sense* [25, 18]. Low effective dimension can often be achieved by redefining f without changing its expectation, via a change of variables [26, 25, 27, 28, 14, 29, 30]. That is, we change the way the uniforms (the coordinates of $\mathbf{u}$) are transformed in the simulation. There are important applications in computational finance, for example, where after such a change of variable, the one- and two-dimensional functions $f_{\mathfrak{u}}$ account for more than 99% of the variability of f [14, 30]. For these applications, it is important that the one- and two-dimensional projections $P_n(\mathfrak{u})$ have very good uniformity, and there is no need to care much about the high-dimensional projections.

It makes sense that the figures of merit for the point sets Ψ_s produced by RNGs also take the low-dimensional projections into account, as suggested in [3, 31, 17], for example. In fact, the standardized figures of merit based on the spectral test, as defined in [32–34] for example, already do this to a certain extent by giving more weight to the low-dimensional projections in the truncation sense (where $\mathfrak{u} = \{1, \ldots, d\}$ for small d).

2 Examples of Discrepancies for QMC

Discrepancies and the corresponding variations are often defined via reproducing kernel Hilbert spaces (RKHS). An RKHS is constructed by selecting a *kernel* $K : [0,1]^{2s} \to \mathbb{R}$, which is a symmetric and positive semi-definite function. The kernel determines in turn a set of basis functions and a scalar product, that define a Hilbert space $\mathcal{H}$ [35]. For $f \in \mathcal{H}$ and a point set P_n, (2) holds with

$V(f) = \|f - \mu\|_K$, where $\|\cdot\|_K$ is the norm that corresponds to the scalar product of $\mathcal{H}$, and $D(P_n)$ that satisfies

$$D^2(P_n) = \frac{1}{n^2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} K(\mathbf{u}_i,\mathbf{u}_j) - \frac{2}{n}\sum_{i=0}^{n-1}\int_{[0,1]^s} K(\mathbf{u}_i,\mathbf{v})d\mathbf{v} + \int_{[0,1]^{2s}} K(\mathbf{u},\mathbf{v})d\mathbf{u}d\mathbf{v} \quad (4)$$

(see [11, 12]). When $K(\mathbf{u},\mathbf{v})$ can be computed in $O(s)$ time for an arbitrary $(\mathbf{u},\mathbf{v})$, then this $D(P_n)$ can be computed in $O(n^2 s)$ time, but this assumption does not always hold.

One important type of kernel has the form

$$K(\mathbf{u},\mathbf{v}) = \sum_{\mathbf{h}\in\mathbb{Z}^s} w(\mathbf{h}) e^{2\pi\iota\mathbf{h}^{\mathsf{t}}(\mathbf{u}-\mathbf{v})} \quad (5)$$

where $\iota = \sqrt{-1}$, the t means "transposed", and the $w(\mathbf{h})$ are non-negative *weights* such that $\sum_{\mathbf{h}\in\mathbb{Z}^s} w(\mathbf{h}) < \infty$. The corresponding square variation is

$$V^2(f) = \sum_{\mathbf{0}\neq\mathbf{h}\in\mathbb{Z}^s} |\hat{f}(\mathbf{h})|^2 / w(\mathbf{h}),$$

where the $\hat{f}(\mathbf{h})$ are the Fourier coefficients of f. The corresponding discrepancy is easily computable only for special shapes of the weights.

For example, if

$$w(\mathbf{h}) = \prod_{j=1}^{s} \min(1, \gamma_j |h_j|^{-2\alpha}), \quad (6)$$

for some integer $\alpha \geq 1$ and positive real numbers (weights) γ_j, then the kernel becomes

$$K_\alpha(\mathbf{u},\mathbf{v}) = \prod_{j=1}^{s}\left[1 - \gamma_j \frac{(-4\pi^2)^\alpha}{(2\alpha)!} B_{2\alpha}((u_j - v_j) \bmod 1)\right], \quad (7)$$

where $B_{2\alpha}$ is the Bernoulli polynomial of degree 2α [20]. This kernel can be computed in $O(s)$ time, so the discrepancy can be computed in $O(n^2 s)$ time. The corresponding $V(f)$ in this case satisfies

$$V^2(f) = \sum_{\phi\neq\mathfrak{u}\subseteq\mathcal{S}} \left(\prod_{j\in\mathfrak{u}} \gamma_j^{-\alpha}\right) (4\pi^2)^{-\alpha|\mathfrak{u}|} \int_{[0,1]^{|\mathfrak{u}|}} \left| \int_{[0,1]^{s-|\mathfrak{u}|}} \frac{\partial^{\alpha|\mathfrak{u}|} f}{\partial \mathbf{u}_{\mathfrak{u}}^{\alpha}}(\mathbf{u}) d\mathbf{u}_{\bar{\mathfrak{u}}} \right|^2 d\mathbf{u}_{\mathfrak{u}}, \quad (8)$$

where $\mathbf{u}_{\mathfrak{u}}$ represents the coordinates of $\mathbf{u}$ whose indices are in the set $\mathfrak{u}$ and $\mathbf{u}_{\bar{\mathfrak{u}}}$ represents those whose indices are not in $\mathfrak{u}$. This RKHS contains only periodic functions f of period 1 with respect to each coordinate. and the variability measures the smoothness via the partial derivatives of f.

Slight variations of this discrepancy have interesting geometric interpretations. For example, if we replace $B_{2\alpha}((u_j - v_j) \bmod 1)$ in (7) by an appropriate

(simple) polynomial in u_j and v_j, we obtain a weighted $\mathcal{L}_2$-*unanchored discrepancy* whose interpretation is the following [11, 12]. For each subset $\mathfrak{u}$ of coordinates and any $\mathbf{u}, \mathbf{v} \in [0,1]^{|\mathfrak{u}|}$, let $D(P_n(\mathfrak{u}), \mathbf{u}, \mathbf{v})$ be the absolute difference between the volume of the $|\mathfrak{u}|$-dimensional box with opposite corners at $\mathbf{u}$ and $\mathbf{v}$, and the fraction of the points of P_n that fall in that box. The square discrepancy is then defined as

$$[D_2(P_n)]^2 = \sum_{\phi \neq \mathfrak{u} \subseteq \mathcal{S}} \left(\prod_{j \in \mathfrak{u}} \gamma_j \right) \int_{[0,1]^{|\mathfrak{u}|}} \int_{[\mathbf{0}, \mathbf{v}]} D^2(P_n(\mathfrak{u}), \mathbf{u}, \mathbf{v}) d\mathbf{u} d\mathbf{v}. \tag{9}$$

Other similarly defined RKHSs contain non-periodic functions f. For example, by taking again the appropriate (simple) function in place of $B_{2\alpha}((u_j - v_j) \bmod 1)$ in (7), we obtain a weighted $\mathcal{L}_2$-*star discrepancy*, whose square can be written as

$$[D_2(P_n)]^2 = \sum_{\phi \neq \mathfrak{u} \subseteq \mathcal{S}} \left(\prod_{j \in \mathfrak{u}} \gamma_j \right) \int_{[0,1]^{|\mathfrak{u}|}} D^2(P_n(\mathfrak{u}), \mathbf{0}, \mathbf{v}) d\mathbf{v} \tag{10}$$

and the square variation is

$$V^2(f) = \sum_{\phi \neq \mathfrak{u} \subseteq \mathcal{S}} \left(\prod_{j \in \mathfrak{u}} \gamma_j^{-1} \right) \int_{[0,1]^{|\mathfrak{u}|}} \left| \frac{\partial^{|\mathfrak{u}|}}{\partial \mathbf{u}_{\mathfrak{u}}} f_{\mathfrak{u}}(\mathbf{u}_{\mathfrak{u}}) \right|^2 d\mathbf{u}_{\mathfrak{u}}$$

(see [11, 12]). All these discrepancies can be computed in $O(n^2 s)$ time for general point sets.

It is known that there exists point sets P_n for which the discrepancy based on (7) converges as as $O(n^{-\alpha+\delta})$ for any $\delta > 0$, and point sets for which the discrepancy (10) converges as $O(n^{-1+\delta})$.

3 RNGs Based on Linear Recurrences Modulo a Large Integer m

An important (and widely used) class of RNGs is based on the general linear recurrence

$$x_i = (a_1 x_{i-1} + \cdots + a_k x_{i-k}) \bmod m, \tag{11}$$

where k and m are positive integers, $a_1, \ldots, a_k \in \{0, 1, \ldots, m-1\}$, and $a_k \neq 0$. The state at step i is $s_i = \mathbf{x}_i = (x_{i-k+1}, \ldots, x_i)$. Suppose the output is $u_i = x_i/m$ (in practice it is slightly modified to make sure that $0 < u_i < 1$). This RNG is called a *multiple recursive generator*. For $k = 1$, it is known as a *linear congruential generator* (LCG). For prime m and well-chosen coefficients, the period length can reach $m^k - 1$ [1], which can be made arbitrarily large by increasing k. Because of the linearity, jumping ahead from $\mathbf{x}_i$ to $\mathbf{x}_{i+\nu}$ is easy,

regardless of ν: One can write $\mathbf{x}_{i+\nu} = \mathbf{A}_\nu \mathbf{x}_i \bmod m$, where $\mathbf{A}_\nu$ is a $k \times k$ matrix that can be precomputed once for all [3].

It is well-known that in this case, Ψ_s is the intersection of a lattice

$$L_s = \left\{ \mathbf{v} = \sum_{j=1}^{s} h_j \mathbf{v}_j \text{ such that each } h_j \in \mathbb{Z} \right\} \tag{12}$$

with the unit hypercube $[0,1)^s$ [1, 36], where a lattice basis $\{\mathbf{v}_1, \ldots, \mathbf{v}_s\}$ is easy to obtain [36]. Theoretical measures of uniformity used in practice are defined in terms of the geometry of this lattice. The lattice structure implies in particular that Ψ_s is contained in a family of equidistant parallel hyperplanes. For the family for which the successive hyperplanes are farthest apart, the inverse of the distance between the successive hyperplanes is equal to the Euclidean length of the shortest nonzero vector in the dual lattice L_s^*, defined by $L_s^* = \{\mathbf{h} \in \mathbb{R}^s : \mathbf{h}^t\mathbf{v} \in \mathbb{Z} \text{ for all } \mathbf{v} \in L_s\}$. The $\mathcal{L}_1$ length of the shortest nonzero vector in L_s^* gives the number of hyperplanes required to cover the points. The computing time of these shortest vectors is exponential in s (in the worst case, with the best known algorithms), but depends very little on m and k. In practice, one can handle values of s up to 50 (or more, for easier lattices), even with $m^k > 2^{1000}$ [32].

Note that for every subset of coordinates $\mathfrak{u} = \{i_1, \ldots, i_d\}$, the projection set $\Psi(\mathfrak{u})$ is also the intersection of a lattice with the unit hypercube $[0,1)^d$, and one can compute the length $\ell(\mathfrak{u})$ of a shortest nonzero vector in the corresponding dual lattice $L^*(\mathfrak{u})$. Moreover, for any given number of points $n = m^k$ and a given dimension d, there is a theoretical upper bound $\ell_d^*(n)$ on this length $\ell(\mathfrak{u})$. One can divide $\ell(\mathfrak{u})$ by such an upper bound to obtain a standardized value between 0 and 1, and take the worst case over a selected class $\mathcal{J}$ of index sets $\mathfrak{u}$. This gives a figure of merit of the form

$$\min_{\mathfrak{u} \in \mathcal{J}} \ell(\mathfrak{u}) / \ell_{|\mathfrak{u}|}^*(n). \tag{13}$$

This type of criterion has been proposed in [17]. Simplified versions, where $\mathcal{J}$ contains only the subsets of successive coordinates up to a given dimension, have been used for a long time to measure the quality of RNGs [34, 1, 32, 3].

It is important to look at the projections $\Psi(\mathfrak{u})$, because fast long-period (but otherwise poorly designed) RNGs often have some very bad low-dimensional projections. For example, *lagged-Fibonacci* generators follow the recurrence (11) with only two nonzero coefficients, say a_r and a_k, both equal to ± 1. It turns out that for these generators, with $\mathfrak{u} = \{0, k-r, k\}$, all the points of $\Psi_s(\mathfrak{u})$ lie in only two parallel planes in the three-dimensional cube. This type of structure can have a disastrous impact on certain simulations. The add-with-carry and subtract-with-borrow generators, proposed in [37] and still available in some popular software, have exactly the same problem. A well-chosen figure of merit of the form (13) should give high penalties to these types of bad projections.

A variant of the multiple recursive generator is the multiply-with-carry generator, based on a linear recurrence with carry:

$$\begin{aligned} x_i &= (a_1 x_{i-1} + \cdots + a_k x_{i-k} + c_{i-1}) d \bmod b, \\ c_i &= \lfloor (a_0 x_i + a_1 x_{i-1} + \cdots + a_k x_{i-k} + c_{i-1})/b \rfloor, \\ u_i &= x_i/b + x_{i+1}/b^2 + \cdots \end{aligned}$$

where $b \geq 2$ is an integer, a_0 is relatively prime to b, and d is the multiplicative inverse of $-a_0$ modulo b. In practice, the expansion that defines u_i is truncated to a few terms. An important result, if we neglect this truncation, states that this RNG is exactly equivalent to an LCG with modulus $m = \sum_{\ell=0}^{k} a_\ell b^\ell$ and multiplier a equal to the inverse of b modulo m [38–40]. This means that this RNG can be seen as a clever way of implementing an LCG with very large modulus and large period (this RNG can be quite fast if b is a power of two, for example), and that its uniformity can be measured in terms of the lattice structure of this LCG [39].

4 RNGs Based on Linear Recurrences Modulo 2

Another important class of construction methods uses a linear recurrence as in (11), but with $m = 2$ [41, 42]. That is, all operations are performed in the finite field $\mathbb{F}_2 = \{0, 1\}$. This general construction can be written in matrix form as [3, 31]:

$$\mathbf{x}_i = \mathbf{A}\mathbf{x}_{i-1}, \quad \mathbf{y}_i = \mathbf{B}\mathbf{x}_i, \quad u_i = \sum_{\ell=1}^{w} y_{i,\ell-1} 2^{-\ell},$$

where $\mathbf{x}_i = (x_{i,0}, \ldots, x_{i,k-1})^\mathsf{t}$ is the *state* at step i, $\mathbf{A}$ is a $k \times k$ binary matrix, $\mathbf{y}_i = (y_{i,0}, \ldots, y_{i,w-1})^\mathsf{t}$, $\mathbf{B}$ is a $w \times k$ binary matrix, k and w are positive integers, and $u_i \in [0, 1)$ is the *output* at step i. (In practice, the output can be modified slightly to avoid returning 0.)

This framework covers several well-known generators such that the Tausworthe, polynomial LCG, generalized feedback shift register (GFSR), twisted GFSR, Mersenne twister, WELL, xorshift, linear cellular automaton, and combinations of these [43, 3, 31, 44, 42]. With a careful design, for which $\mathbf{A}$ has a primitive characteristic polynomial over $\mathbb{F}_2$, the period length can reach $2^k - 1$. The matrices $\mathbf{A}$ and $\mathbf{B}$ should be constructed so that the products (14) and (14) can be computed very quickly by a few simple binary operations on blocks of bits, such as or, exclusive-or, shift, and rotation. A compromise must be made between the number of such operations and a good uniformity of the point sets Ψ_s (with too few operations, there are in general limitations on the quality of the uniformity that can be reached). For these types of generators, the uniformity of the point sets Ψ_s is measured by their *equidistribution* properties, as explained in Section 6.

Combined generators of this type, defined by xoring the output vectors $\mathbf{y}_i$ of the components, are equivalent to yet another generator from the same class.

Such combinations provide efficient ways of implementing RNGs with larger state spaces [45, 46, 31].

5 Lattice rules for QMC

When a lattice L_s as in (12) contains $\mathbb{Z}^s$, it is called an *integration lattice*, and the QMC approximation (1) with $P_n = L_s \cap [0,1)^s$ is a *lattice rule* [4, 20]. The most frequently used lattice rules are of rank 1: we have $\mathbf{v}_1 = \mathbf{a}_1/n$ where $\mathbf{a}_1 = (a_1, \ldots, a_s)$, and $\mathbf{v}_j = \mathbf{e}_j$ (the jth unit vector) for $j \geq 2$. A special case is when P_n is the point set Ψ_s that correspond to a LCG; we then have a Korobov lattice rule.

Integration lattices are usually randomized by a random shift modulo 1. The shift preserves the lattice structure. It turns out that the variance of the corresponding RQMC estimator can be written explicitly as

$$\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}] = \sum_{\mathbf{0} \neq \mathbf{h} \in L_s^*} |\hat{f}(\mathbf{h})|^2, \tag{14}$$

where the $\hat{f}(\mathbf{h})$ are the coefficients in the Fourier expansion of the function f [17]. Assuming that we want to minimize the variance, (14) gives the ultimate discrepancy measure of a lattice point set to integrate a given function f. The square Fourier coefficients are generally unknown, but they can be replaced by weights $w(\mathbf{h})$ that try to approximate their expected behavior for a given class of functions of interest. It might be difficult to obtain such an approximation. Another problem is that computing (14) with the weights $w(\mathbf{h})$, and searching for lattices that minimize its value, may be difficult unless the weights have a special form. In [17], the authors argue that since the Fourier coefficients for the short vectors $\mathbf{h}$ represent the main trends of the function's behavior, they are likely to be those having the most impact on the variance, so we should try to keep them out of the dual lattice to eliminate them from the sum in (14). If we do this for a selected class of low-dimensional projections of L_s, and weight these projections, this leads to the same criterion as in (13). Specific Korobov lattice parameters found on the basis of this criterion are given in [17].

In these criteria, the Euclidean length of $\mathbf{h}$ could also be replaced by other notions of length; for example, the $\mathcal{L}_1$ length, as discussed earlier, of the product length $\prod_{j=1}^{s} \max(1, |h_j|)$, for which the length of the shortest vector in L_s^* is called the *Zaremba index* [20].

For lattice rules, discrepancies based on kernels of the general form (5) admit simplified formulas, thanks to the fact that $\sum_{i=0}^{n-1} e^{2\pi\iota \mathbf{h}^{\mathrm{t}}\mathbf{u}_i} = n$ for $\mathbf{h} \in L_s^*$, and 0 otherwise [20]. In particular, the square discrepancy for the kernel (7) simplifies to

$$D^2(P_n) = -1 + \frac{1}{n} \sum_{i=0}^{n-1} \prod_{j=1}^{s} \left(1 + \gamma_j \frac{(-4\pi^2)^\alpha}{(2\alpha)!} B_{2\alpha}(u_j)/2\right), \tag{15}$$

which can be computed in $O(ns)$ time. This discrepancy is a weighted version of a criterion known as $P_{2\alpha}$ [11, 20], widely used for lattice rules.

It is known that for any given dimension s and arbitrarily small $\delta > 0$, there exist lattice rules whose discrepancy (15) is $O(n^{-\alpha+\delta})$ [47, 4, 20]. Until very recently, it was unclear if those lattices were easy to find, but explicit construction methods are now available.

Indeed, for a large s and moderate n, trying all possibilities for the basis vectors is just impractical. Even if we restrict ourselves to a rank-1 lattice, there is already $(n-1)^s$ possibilities. So one must either search at random, or perform a more restricted search. One possibility is to limit the search to Korobov rules, so that there is only the parameter a_1 to select. Another possibility is to adopt a greedy component-by-component (CBC) construction of a rank-1 lattice, for a given n, by selecting the components a_j of the vector $\mathbf{a}_1 = (a_1, \ldots, a_s)$ iteratively as follows [48, 49]. Start with $a_1 = 1$. At step j, $a_1, \ldots, a_{j-1}$ are fixed and we select a_j to optimize a given discrepancy measure for the j-dimensional rank-1 lattice with generating vector $\mathbf{v}_1 = \mathbf{a}_1/n$. At each step, there is at most $n-1$ choices to examine for a_j, so at most $O(ns)$ discrepancies need to be computed to construct a lattice of dimension s. For a discrepancy of the form (15), since each discrepancy is computable in $O(ns)$ time, we can conclude that computing $\mathbf{a}_1 = (a_1, \ldots, a_s)$ requires at most $O(n^2 s^2)$ time. But in fact, faster algorithms have been designed that can compute $\mathbf{a}_1$ in $O(n \log(n) s)$ time using $O(n)$ memory [50, 48, 51].

The remarkable feature of these CBC constructions is that for a large variety of discrepancies, defined mostly via RKHS, it has been proved that one obtains rank-1 lattices whose discrepancy converges at the same rate, as a function of n, as the best possible lattice constructions [50, 52, 47, 53]. In other words, for these discrepancies, CBC provides a practical way of constructing lattices with optimal convergence rate of the discrepancy. These results provide supporting arguments for the use of these types of discrepancies as figures of merit.

6 Digital nets for QMC

We start with an arbitrary integer $b \geq 2$, usually a prime. An s-dimensional *digital net in base* b, with $n = b^k$ points, is a point set $P_n = \{\mathbf{u}_0, \ldots, \mathbf{u}_{n-1}\}$ defined by selecting s generator matrices $\mathbf{C}_1, \ldots, \mathbf{C}_s$ with elements in $\mathbb{Z}_b = \{0, \ldots, b-1\}$, where each $\mathbf{C}_j$ is $\infty \times k$. The points $\mathbf{u}_i$ are defined as follows. For $i = 0, \ldots, b^k - 1$, we write

$$i = \sum_{\ell=0}^{k-1} a_{i,\ell} b^\ell,$$

$$\begin{pmatrix} u_{i,j,1} \\ u_{i,j,2} \\ \vdots \end{pmatrix} = \mathbf{C}_j \begin{pmatrix} a_{i,0} \\ a_{i,1} \\ \vdots \\ a_{i,k-1} \end{pmatrix} \bmod b, \tag{16}$$

$$u_{i,j} = \sum_{\ell=1}^{\infty} u_{i,j,\ell} b^{-\ell}, \quad \text{and} \quad \mathbf{u}_i = (u_{i,1}, \ldots, u_{i,s}). \tag{17}$$

In practical implementations, only the first r rows of the $\mathbf{C}_j$'s are nonzero, for some positive integer r (for example, with $b^r \approx 2^{31}$ on 32-bit computers).

It is usually the case that the first k rows of each $\mathbf{C}_j$ form a nonsingular $k \times k$ matrix, so each one-dimensional projection $P_n(\{j\})$ truncated to the first k digits is equal to $\mathbb{Z}_n/n = \{0, 1/n, \ldots, (n-1)/n\}$. The role of each $\mathbf{C}_j$ is to define a permutation of $\mathbb{Z}_n/n$ so that these numbers are enumerated in a different order for the different coordinates. The choice of these permutations determines the uniformity of P_n and of its projections $P_n(\mathfrak{u})$ (which are also digital nets).

In a more general definition of digital net [4], one can also apply bijections (or permutations) to the digits of $\mathbb{Z}_b$ before and after the multiplication by $\mathbf{C}_j$. These multiplications are performed in an arbitrary ring R of cardinality b, and the bijections may depend on ℓ and j. The bijections provide additional opportunity for improving the uniformity.

A *digital sequence in base b* is defined by selecting matrices $\mathbf{C}_j$ with an infinite number of columns. This gives an infinite sequence of points. For each k, the first k columns determine the first b^k points, which form a digital net. Widely-used instances of digital sequences are those of Sobol' [54] in base 2, of Faure [55] in prime base b, of Niederreiter [56], and of Niederreiter and Xing [57]. With an infinite sequence of matrices $\mathbf{C}_j$, we have an infinite-dimensional digital net. These infinite sequences of columns and matrices are often defined via recurrences (each column and matrix being a function of the previous ones).

Polynomial lattice rules use point sets defined by a lattice as in (12), but where the h_j are polynomials over $\mathbb{Z}_b$, and the coordinates of the $\mathbf{v}_j$ are polynomials over $\mathbb{Z}_b$ divided by a common polynomial of degree k. The output is produced simply by "evaluating" each $\mathbf{v}(z)$, which is a vector of formal series in z, at $z = b$. These polynomial lattice rules turn out to be special cases of digital nets in base b. Moreover, much of the theory developed for ordinary lattice rules has a counterpart for those other types of lattice rules [58, 59].

Important parts of this theory also extends to digital nets in general [60, 61]. In particular, there is a dual space $\mathcal{C}_s^*$ that plays the same role as the dual lattice L_s^* (in the case of polynomial lattice rules, $\mathcal{C}_s^*$ is also a lattice). For a digital net with a random digital shift in base b, in analogy with (14), the RQMC estimator has variance

$$\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}] = \sum_{\mathbf{0} \neq \mathbf{h} \in \mathcal{C}_s^*} |\tilde{f}(\mathbf{h})|^2, \tag{18}$$

where the $\tilde{f}(\mathbf{h})$ are the coefficients of the Walsh expansion of f. For a given f, this expression provides an ultimate discrepancy measure for a digital net with a random digital shift. This discrepancy has the same limitations as (14) for ordinary lattices (the Walsh coefficients are usually unknown, etc), and the practical alternatives are analogous. They lead to figures of merit as in (13), but where the $\ell(\mathfrak{u})$ are lengths of shortest vectors in the dual spaces $\mathcal{C}^*(\mathfrak{u})$ associated with the projections $P_n(\mathfrak{u})$ instead of in the dual lattices $L^*(\mathfrak{u})$.

Do the discrepancies discussed in Section 2 have simplified expressions for digital nets, as was the case for lattice rules? Those based on the kernel (5) do not, but they do if we take a slightly different kernel, based on Walsh expansions

in base b instead of Fourier expansions. This can be exploited to develop practical figures of merit analogous to those used for lattice rules.

The most widely used class of figures of merit, for both RNG and QMC point sets, are those based on the notion of *equidistribution*, defined as follows. For a vector of non-negative integers $(q_1, \ldots, q_s)$, we partition the jth axis into b^{q_j} equal parts for each j. This partitions the hypercube $[0,1)^s$ into $b^{q_1+\cdots+q_s}$ rectangular boxes of the same size and shape. A point set P_n of cardinality $n = b^k$ is $(q_1, \ldots, q_s)$-*equidistributed in base* b if each box of this partition contains exactly b^t points of P_n, where $t = k - q_1 - \cdots - q_s$. For a digital net in base b, this property holds if and only if the set of $k - q = q_1 + \cdots + q_s$ rows that comprise the first q_j rows of $\mathbf{C}_j$, for $j = 1, \ldots, s$, is linearly independent in the finite ring R [4].

The set P_n is a (t, k, s)-*net in base* b if it is $(q_1, \ldots, q_s)$-equidistributed whenever $q_1 + \cdots + q_s \le k - t$ [4]. The smallest such t is the t-*value* of the net. A digital sequence $\{\mathbf{u}_0, \mathbf{u}_1, \ldots, \}$ in s dimensions is a (t, s)-*sequence in base* b if for all integers $k > 0$ and $\nu \ge 0$, the point set $Q(k, \nu) = \{\mathbf{u}_i : i = \nu b^k, \ldots, (\nu+1)b^k - 1\}$ is a (t, k, s)-net in base b. The t-value is a widely used figure of merit for digital nets. Ideally, we would like it to be zero, but there are theoretical lower bounds on it. In particular, a $(0, k, s)$-net in base b can exist only if $b \ge s - 1$, and a $(0, s)$-sequence in base b can exist only if $b \ge t$. Lower bounds for general pairs (b, s), together with the best values achieved by known constructions, are tabulated in [62]. For example, for $b = 2$, $k = 16$, and $s = 20$, the t-value cannot be smaller than 9. A t-value of 9 in this case only guarantees equidistribution for a partition in $2^7 = 128$ boxes, which must contain $2^9 = 512$ points each. For a given partition, this is a weak requirement.

The problem is that a small t-value would require equidistribution for a very rich family of partitions into rectangular boxes, and this becomes impossible when t is too small. To get around this, we may restrict our consideration to a smaller family of partitions; for example, only cubic boxes. We then want P_n to be $(\ell, \ldots, \ell)$-equidistributed for the largest possible ℓ, which obviously cannot exceed $\lfloor k/s \rfloor$. We want to minimize the *resolution gap* $\delta = \lfloor k/s \rfloor - \ell$.

These definitions apply to the projections $P_n(\mathfrak{u})$ as well. Let $t_\mathfrak{u}$ and $\delta_\mathfrak{u}$ denote the t-value and the resolution gap for $P_n(\mathfrak{u})$, and $t^*_{|\mathfrak{u}|}$ a theoretical lower bound on $t_\mathfrak{u}$. Discrepancy measures for digital nets can be defined by

$$\max_{\mathfrak{u}\in\mathcal{J}} \gamma_\mathfrak{u}\delta_\mathfrak{u}, \quad \text{or} \quad \sum_{\mathfrak{u}\in\mathcal{J}} \gamma_\mathfrak{u}\delta_\mathfrak{u}, \quad \text{or}$$

$$\max_{\mathfrak{u}\in\mathcal{J}} \gamma_\mathfrak{u}\left[t_\mathfrak{u} - t^*_{|\mathfrak{u}|}\right], \quad \text{or} \quad \sum_{\mathfrak{u}\in\mathcal{J}} \gamma_\mathfrak{u}\left[t_\mathfrak{u} - t^*_{|\mathfrak{u}|}\right],$$

for some non-negative weights $\gamma_\mathfrak{u}$ and a preselected class $\mathcal{J}$ of index sets $\mathfrak{u}$ [58, 13, 59, 63]. The choice of $\mathcal{J}$ is a matter of compromise. With a larger (richer) $\mathcal{J}$, the criterion is more expensive to compute, and its best possible value is generally larger, so it will have less discriminating power for the more important low-dimensional projections. In practice, the weights are often taken all equal to 1.

Specific instances of these criteria, and search results for good parameters for specific types of digital nets, are reported in [64, 21, 65, 63]. A special case of the first of these four criteria has been widely used to assess the uniformity of $\mathbb{F}_2$-linear RNGs [45, 66, 31, 67, 10, 42].

7 Conclusion

Low-discrepancy point sets and sequences used for QMC, and the point sets formed by vectors of successive output values produced by RNGs, have much in common. They are often defined via similar linear recurrences. We also want both of them to be highly uniform in the unit hypercube. However, the figures of merit commonly used to measure their uniformity are slightly different. One of the reasons for this difference is the difference of cardinality between those types of point sets: For RNGs, the cardinality is huge and we must restrict ourselves to criteria that can be computed without enumerating the points explicitly, for example. For QMC, on the other hand, certain discrepancies are motivated by the fact that they provide explicit error bounds or variance bounds on certain classes of functions.

References

1. Knuth, D.E.: The Art of Computer Programming, Volume 2: Seminumerical Algorithms. Third edn. Addison-Wesley, Reading, MA (1998)
2. L'Ecuyer, P.: Uniform random number generation. Annals of Operations Research **53** (1994) 77–120
3. L'Ecuyer, P.: Uniform random number generation. In Henderson, S.G., Nelson, B.L., eds.: Simulation. Handbooks in Operations Research and Management Science. Elsevier, Amsterdam, The Netherlands (2006) 55–81 Chapter 3.
4. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods. Volume 63 of SIAM CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, Philadelphia, PA (1992)
5. Law, A.M., Kelton, W.D.: Simulation Modeling and Analysis. Third edn. McGraw-Hill, New York, NY (2000)
6. L'Ecuyer, P., Buist, E.: Simulation in Java with SSJ. In Kuhl, M.E., Steiger, N.M., Armstrong, F.B., Joines, J.A., eds.: Proceedings of the 2005 Winter Simulation Conference, Pistacaway, NJ, IEEE Press (2005) 611–620
7. L'Ecuyer, P.: Pseudorandom number generators. In Platen, E., Jaeckel, P., eds.: Simulation Methods in Financial Engineering. Encyclopedia of Quantitative Finance. Wiley (2008) Forthcoming.
8. Deng, L.Y.: Efficient and portable multiple recursive generators of large order. ACM Transactions on Modeling and Computer Simulation **15**(1) (2005) 1–13
9. L'Ecuyer, P., Simard, R.: TestU01: A C library for empirical testing of random number generators. ACM Transactions on Mathematical Software **33**(4) (August 2007) Article 22
10. Panneton, F., L'Ecuyer, P., Matsumoto, M.: Improved long-period generators based on linear recurrences modulo 2. ACM Transactions on Mathematical Software **32**(1) (2006) 1–16

11. Hickernell, F.J.: A generalized discrepancy and quadrature error bound. Mathematics of Computation **67** (1998) 299–322
12. Hickernell, F.J.: What affects the accuracy of quasi-Monte Carlo quadrature? In Niederreiter, H., Spanier, J., eds.: Monte Carlo and Quasi-Monte Carlo Methods 1998, Berlin, Springer-Verlag (2000) 16–55
13. L'Ecuyer, P., Lemieux, C.: Recent advances in randomized quasi-Monte Carlo methods. In Dror, M., L'Ecuyer, P., Szidarovszky, F., eds.: Modeling Uncertainty: An Examination of Stochastic Theory, Methods, and Applications. Kluwer Academic, Boston (2002) 419–474
14. L'Ecuyer, P., Lécot, C., Tuffin, B.: A randomized quasi-Monte Carlo simulation method for Markov chains. Operations Research (2008) To appear.
15. Hickernell, F.J., Sloan, I.H., Wasilkowski, G.W.: On strong tractability of weighted multivariate integration. Mathematics of Computation **73**(248) (2004) 1903–1911
16. Ben-Ameur, H., L'Ecuyer, P., Lemieux, C.: Combination of general antithetic transformations and control variables. Mathematics of Operations Research **29**(4) (2004) 946–960
17. L'Ecuyer, P., Lemieux, C.: Variance reduction via lattice rules. Management Science **46**(9) (2000) 1214–1235
18. Owen, A.B.: Latin supercube sampling for very high-dimensional simulations. ACM Transactions on Modeling and Computer Simulation **8**(1) (1998) 71–102
19. Cranley, R., Patterson, T.N.L.: Randomization of number theoretic methods for multiple integration. SIAM Journal on Numerical Analysis **13**(6) (1976) 904–914
20. Sloan, I.H., Joe, S.: Lattice Methods for Multiple Integration. Clarendon Press, Oxford (1994)
21. L'Ecuyer, P., Lemieux, C.: Quasi-Monte Carlo via linear shift-register sequences. In: Proceedings of the 1999 Winter Simulation Conference, IEEE Press (1999) 632–639
22. Matoušek, J.: Geometric Discrepancy: An Illustrated Guide. Springer-Verlag, Berlin (1999)
23. Liu, R., Owen, A.B.: Estimating mean dimensionality. manuscript (2003)
24. Wang, X., Sloan, I.H.: Why are high-dimensional finance problems often of low effective dimension? SIAM Journal on Scientific Computing **27**(1) (2005) 159–183
25. Caflisch, R.E., Morokoff, W., Owen, A.: Valuation of mortgage-backed securities using Brownian bridges to reduce effective dimension. The Journal of Computational Finance **1**(1) (1997) 27–46
26. Avramidis, T., L'Ecuyer, P.: Efficient Monte Carlo and quasi-Monte Carlo option pricing under the variance-gamma model. Management Science **52**(12) (2006) 1930–1944
27. Glasserman, P.: Monte Carlo Methods in Financial Engineering. Springer-Verlag, New York (2004)
28. Imai, J., Tan, K.S.: A general dimension reduction technique for derivative pricing. Journal of Computational Finance **10**(2) (2006) 129–155
29. Morokoff, W.J.: Generating quasi-random paths for stochastic processes. SIAM Review **40**(4) (1998) 765–788
30. Wang, X., Sloan, I.H.: Brownian bridge and principal component analysis: Toward removing the curse of dimensionality. IMA Journal of Numerical Analysis **27** (2007) 631–654
31. L'Ecuyer, P., Panneton, F.: $\mathbf{F}_2$-linear random number generators. In Alexopoulos, C., Goldsman, D., eds.: Advancing the Frontiers of Simulation: A Festschrift in Honor of George S. Fishman. Springer-Verlag, New York (2007) To appear.

32. L'Ecuyer, P.: Good parameters and implementations for combined multiple recursive random number generators. Operations Research **47**(1) (1999) 159–164
33. L'Ecuyer, P.: Tables of linear congruential generators of different sizes and good lattice structure. Mathematics of Computation **68**(225) (1999) 249–260
34. Fishman, G.S.: Monte Carlo: Concepts, Algorithms, and Applications. Springer Series in Operations Research. Springer-Verlag, New York, NY (1996)
35. Wahba, G.: Spline Models for Observational Data. Volume 59 of SIAM CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, Philadelphia, PA (1990)
36. L'Ecuyer, P., Couture, R.: An implementation of the lattice and spectral tests for multiple recursive linear random number generators. INFORMS Journal on Computing **9**(2) (1997) 206–217
37. Marsaglia, G., Zaman, A.: A new class of random number generators. The Annals of Applied Probability **1** (1991) 462–480
38. Tezuka, S., L'Ecuyer, P., Couture, R.: On the add-with-carry and subtract-with-borrow random number generators. ACM Transactions of Modeling and Computer Simulation **3**(4) (1994) 315–331
39. Couture, R., L'Ecuyer, P.: Distribution properties of multiply-with-carry random number generators. Mathematics of Computation **66**(218) (1997) 591–607
40. Goresky, M., Klapper, A.: Efficient multiply-with-carry random number generators with maximal period. ACM Transactions on Modeling and Computer Simulation **13**(4) (2003) 310–321
41. Golomb, S.W.: Shift-Register Sequences. Holden-Day, San Francisco (1967)
42. Tezuka, S.: Uniform Random Numbers: Theory and Practice. Kluwer Academic Publishers, Norwell, MA (1995)
43. L'Ecuyer, P.: Tables of maximally equidistributed combined LFSR generators. Mathematics of Computation **68**(225) (1999) 261–269
44. Matsumoto, M., Nishimura, T.: Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions on Modeling and Computer Simulation **8**(1) (1998) 3–30
45. L'Ecuyer, P.: Maximally equidistributed combined Tausworthe generators. Mathematics of Computation **65**(213) (1996) 203–213
46. L'Ecuyer, P., Panneton, F.: A new class of linear feedback shift register generators. In Joines, J.A., Barton, R.R., Kang, K., Fishwick, P.A., eds.: Proceedings of the 2000 Winter Simulation Conference, Pistacaway, NJ, IEEE Press (2000) 690–696
47. Dick, J., Sloan, I.H., Wang, X., Wozniakowski, H.: Good lattice rules in weighted Korobov spaces with general weights. Numerische Mathematik **103** (2006) 63–97
48. Nuyens, D., Cools, R.: Fast algorithms for component-by-component construction of rank-1 lattice rules in shift-invariant reproducing kernel Hilbert spaces. Mathematics and Computers in Simulation **75** (2006) 903–920
49. Sloan, I.H., Kuo, F.Y., Joe, S.: On the step-by-step construction of quasi-Monte Carlo rules that achieve strong tractability error bounds in weighted Sobolev spaces. Mathematics of Computation **71** (2002) 1609–1640
50. Cools, R., Kuo, F.Y., Nuyens, D.: Constructing embedded lattice rules for multivariate integration. SIAM Journal on Scientific Computing **28**(16) (2006) 2162–2188
51. Nuyens, D., Cools, R.: Fast component-by-component construction of rank-1 lattice rules with a non-prime number of points. Journal of Complexity **22** (2006) 4–28
52. Dick, J., Sloan, I.H., Wang, X., Wozniakowski, H.: Liberating the weights. Journal of Complexity **20**(5) (2004) 593–623

53. Kuo, F.Y., Sloan, I.H.: Lifting the curse of dimensionality. Notices of the AMS **52**(11) (2005) 1320–1328
54. Sobol', I.M.: The distribution of points in a cube and the approximate evaluation of integrals. U.S.S.R. Comput. Math. and Math. Phys. **7** (1967) 86–112
55. Faure, H.: Discrépance des suites associées à un système de numération en dimension s. Acta Arithmetica **61** (1982) 337–351
56. Niederreiter, H.: Point sets and sequences with small discrepancy. Monatshefte für Mathematik **104** (1987) 273–337
57. Niederreiter, H., Xing, C.: The algebraic-geometry approach to low-discrepancy sequences. In Hellekalek, P., Larcher, G., Niederreiter, H., Zinterhof, P., eds.: Monte Carlo and Quasi-Monte Carlo Methods 1996. Volume 127 of Lecture Notes in Statistics., New York, NY, Springer-Verlag (1998) 139–160
58. L'Ecuyer, P.: Polynomial integration lattices. In Niederreiter, H., ed.: Monte Carlo and Quasi-Monte Carlo Methods 2002, Berlin, Springer-Verlag (2004) 73–98
59. Lemieux, C., L'Ecuyer, P.: Randomized polynomial lattice rules for multivariate integration and simulation. SIAM Journal on Scientific Computing **24**(5) (2003) 1768–1789
60. L'Ecuyer, P., Touzin, R.: On the Deng-Lin random number generators and related methods. Statistics and Computing **14** (2004) 5–9
61. Niederreiter, H., Pirsic, G.: Duality for digital nets and its applications. Acta Arithmetica **97** (2001) 173–182
62. Schmid, W.C., Schürer, R.: MinT, the database for optimal (t, m, s)-net parameters. `http://mint.sbg.ac.at` (2005)
63. Panneton, F., L'Ecuyer, P.: Infinite-dimensional highly-uniform point sets defined via linear recurrences in $\mathbb{F}_{2^w}$. In Niederreiter, H., Talay, D., eds.: Monte Carlo and Quasi-Monte Carlo Methods 2004, Berlin, Springer-Verlag (2006) 419–429
64. Joe, S., Kuo, F.Y.: Constructing Sobol sequences with better two-dimensional projections. SIAM Journal on Scientific Computing (2008) to appear.
65. Lemieux, C.: L'utilisation de règles de réseau en simulation comme technique de réduction de la variance. PhD thesis, Université de Montréal (May 2000)
66. L'Ecuyer, P., Panneton, F.: Fast random number generators based on linear recurrences modulo 2: Overview and comparison. In: Proceedings of the 2005 Winter Simulation Conference, IEEE Press (2005) 110–119
67. Panneton, F., L'Ecuyer, P.: On the xorshift random number generators. ACM Transactions on Modeling and Computer Simulation **15**(4) (2005) 346–361