A Tool for Custom Construction of QMC and RQMC Point Sets

Pierre L'Ecuyer, Pierre Marion, Maxime Godin, and Florian Puchhammer

Abstract We present LatNet Builder, a software tool to find good parameters for lattice rules, polynomial lattice rules, and digital nets in base 2, for quasi-Monte Carlo (QMC) and randomized quasi-Monte Carlo (RQMC) sampling over the s-dimensional unit hypercube. The selection criteria are figures of merit that give different weights to different subsets of coordinates. They are upper bounds on the worst-case error (for QMC) or variance (for RQMC) for integrands rescaled to have a norm of at most one in certain Hilbert spaces of functions. We summarize what are the various Hilbert spaces, discrepancies, types of weights, figures of merit, types of constructions, and search methods supported by LatNet Builder. We briefly discuss its organization and we provide simple illustrations of what it can do.

1 Introduction

QMC methods approximate an integral of the form

$$\mu = \int_0^1 \cdots \int_0^1 f(u_1,\ldots,u_s)\,du_1\cdots du_s = \int_{(0,1)^s} f(\mathbf{u})\,d\mathbf{u} = \mathbb{E}[f(\mathbf{U})] \qquad (1)$$

Pierre L'Ecuyer
Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, Canada, e-mail: lecuyer@iro.umontreal.ca

Pierre Marion
Sorbonne Université, CNRS, Laboratoire de Probabilités, Statistique et Modélisation, LPSM, F-75005 Paris, France, e-mail: pierre.marion@upmc.fr

Maxime Godin
Institut des Actuaires, France, e-mail: maxime.godin@institutdesactuaires.com

Florian Puchhammer
Université de Montréal, Canada, and Basque Center for Applied Mathematics, Spain, e-mail: fpuchhammer@bcamath.org

where $f : (0,1)^s \to \mathbb{R}$ and $\mathbf{U}$ is a uniform random vector over $[0,1)^s$, by the average

$$\bar{\mu}_n = \frac{1}{n}\sum_{i=0}^{n-1} f(\mathbf{u}_i) \tag{2}$$

where $P_n = \{\mathbf{u}_0, \ldots, \mathbf{u}_{n-1}\} \subset [0,1)^s$ is a set of n deterministic points that cover the unit hypercube more evenly than typical independent random points. That is, the discrepancy between their empirical distribution and the uniform distribution over $[0,1)^s$ is smaller than for independent random points and converges to 0 faster than $\mathcal{O}(n^{-1/2})$ when $n \to \infty$. This discrepancy can be defined in many ways. It usually represents the worst-case integration error for a given class of integrands f. Typically, this class is a reproducing-kernel Hilbert space (RKHS) $\mathscr{H}$ of functions, such that

$$|E_n| := |\bar{\mu}_n - \mu| \leq \mathscr{D}(P_n)\mathscr{V}(f) \tag{3}$$

for all $f \in \mathscr{H}$, where $\mathscr{V}(f)$ is the norm of $f - \mu$ in $\mathscr{H}$ (we call it the *variation* of f) and $\mathscr{D}(\cdot)$ is the *discrepancy* measure associated with this Hilbert space [8, 19, 40]. For a fixed $f \in \mathscr{H}$ with $\mathscr{V}(f) > 0$, the error bound in (3) converges at the same rate as $\mathscr{D}(P_n)$. A traditional version of (3), whose derivation does not involve Hilbert spaces, is the classical Koksma-Hlawka inequality, in which $\mathscr{V}(f)$ is the Hardy-Krause variation and $\mathscr{D}(P_n)$ is the star discrepancy, which converges as $\mathcal{O}((\log n)^{s-1} n^{-1})$ for well-selected point sets [40]. Another important choice for $\mathscr{H}$ is a Sobolev space of functions whose mixed partial derivatives of order up to α are square-integrable. It is known that for this space, one can construct point sets whose discrepancy converges as $\mathcal{O}((\log n)^{(s-1)/2} n^{-\alpha})$, and that this is the best possible rate [4, 8, 15, 16, 17, 18]. The main classes of QMC point sets are lattice points and digital nets.

For RQMC, the n QMC points are randomized to provide a set of random points $\{\mathbf{U}_0, \ldots, \mathbf{U}_{n-1}\} \subset (0,1)^s$ for which (i) each $\mathbf{U}_i$ individually has the uniform distribution over $[0,1)^s$, and (ii) the points keep their highly-uniform distribution collectively. Randomizations that provably preserve the low discrepancy generally depend on the type of QMC construction: some are used for lattice points and others for digital nets. In some cases, the randomization may even improve the convergence rate of the mean square discrepancy. The RQMC estimator

$$\hat{\mu}_{n,\mathrm{rqmc}} = \frac{1}{n}\sum_{i=0}^{n-1} f(\mathbf{U}_i), \tag{4}$$

which is now random, is unbiased for μ and one wishes to minimize its variance. For more details on RQMC, see for example [23, 27, 30, 31, 38, 45, 46, 47].

The aim of this paper is to introduce *LatNet Builder*, a software tool designed to construct good lattice and digital point sets for QMC and RQMC, in any number of dimensions, for an arbitrary number of points, arbitrary weights on the subsets of coordinates, arbitrary smoothness of the integrands, a variety of construction and randomization methods, and several choices of discrepancies. The point sets can also be extensible in the number of points and number of dimensions. By "constructing

the points" here we mean *defining* the set P_n by selecting the parameter values for a general structure, by trying to miminize a *figure of merit* (FOM) that may represent a discrepancy $\mathscr{D}(P_n)$ or be an upper bound on it. Once this is done, other software can be used to randomize and generate the points for their utilization in applications; see [28, 43] for example. LatNet Builder is available in open source at https://github.com/umontreal-simul/latnetbuilder. It is a descendant of *Lattice Builder* [34], whose scope was limited to ordinary lattice rules. Another related tool is Nuyens' fast CBC constructions [41].

The rest of this paper is organized as follows. In Section 2, we recall the types of QMC point sets covered by our software, namely ordinary lattice points, polynomial lattice points, digital nets, and their higher-order and interlaced versions, as well as the main randomization methods to turn these point sets into RQMC points. The discrepancies that we consider often provide upper bounds on the mean square integration error when using these randomizations, for certain classes of functions. In Section 3, we give the general form of weighted RKHS used in this paper and the corresponding generalized Koksma-Hlawka inequality. We also recall the common types of weights, all supported by the software. In Section 4, we review and justify the various discrepancies that are supported by LatNet Builder and can be used as FOMs to select the parameters of point set constructions. In Section 5, we summarize the search methods implemented in our software. In Section 7, we compare FOM values obtained by various point set constructions and search methods. We also compare RQMC variance for simple integrands f. Section 8 gives a conclusion.

2 Point Set Constructions and Randomizations

LatNet Builder handles ordinary rank-1 lattice points as well as digital nets, which include polynomial lattice rules and high-order and interlaced constructions.

For a *rank-1 lattice rule*, the point set is

$$P_n = \{\mathbf{u}_i = i\mathbf{v}_1 \bmod 1, i = 0, \ldots, n-1\}$$

where $n\mathbf{v}_1 = \mathbf{a} = (a_1, \ldots, a_s) \in \mathbb{Z}_n^s \equiv \{0, \cdots, n-1\}^s$. It is a *Korobov rule* if $\mathbf{a} = (1, a, a^2 \bmod n, \ldots, a^{s-1} \bmod n)$ for some integer $a \in \mathbb{Z}_n$. The parameter to select here is the vector $\mathbf{a}$, for any given n. The usual way to turn a lattice rule into an RQMC point set is by a random shift: generate a single random point $\mathbf{U}$ uniformly in $(0,1)^s$, and add it to each point of P_n, modulo 1, coordinate-wise. This satisfies the RQMC conditions. For more details on lattice rules and their randomly-shifted versions, see [20, 21, 27, 30, 33, 50].

The *Digital nets in base 2* handled by LatNet Builder are defined as follows. The number of points is $n = 2^k$ for some integer k. We select an integer $w \geq k$ and s generating matrices $\mathbf{C}_1, \ldots, \mathbf{C}_s$ of dimensions $w \times k$ and of rank k, with elements in $\mathbb{Z}_2 \equiv \{0,1\}$. The points $\mathbf{u}_i$, $i = 0, \ldots, 2^k - 1$, are defined as follows: for $i = a_{i,0} + a_{i,1}2 + \cdots + a_{i,k-1}2^{k-1}$, we take

$$\begin{pmatrix} u_{i,j,1} \\ \vdots \\ u_{i,j,w} \end{pmatrix} = \mathbf{C}_j \begin{pmatrix} a_{i,0} \\ \vdots \\ a_{i,k-1} \end{pmatrix} \bmod 2, \qquad u_{i,j} = \sum_{\ell=1}^{w} u_{i,j,\ell} 2^{-\ell},$$

and $\mathbf{u}_i = (u_{i,1}, \ldots, u_{i,s})$. There are more general definitions in [8, 40]. The parameters to optimize are the elements of the matrices $\mathbf{C}_j$. Since each $\mathbf{C}_j$ has rank k, each one-dimensional projection truncated to its first k digits is $\mathbb{Z}_n/n = \{0, 1/n, \ldots, (n-1)/n\}$. The ordinary digital nets constructed by LatNet Builder often have $w = k$, so the points have only k digits, but this is not always true.

The most popular digital net constructions are still the *Sobol' points* [52], in base $b = 2$, with $k \times k$ generating matrices that are upper triangular and invertible. These matrices are constructed by a specific method, but the bits of the first few columns above the diagonal can be selected arbitrarily, and their choice has an impact on the quality of the net. General-purpose choices have been proposed in [25, 35], e.g., based on the uniformity of two-dimensional projections. LatNet Builder allows one to construct the matrices based on a much more flexible class of criteria.

A *polynomial lattice rule* (PLR) in base 2 with $n = 2^k$ points is defined as follows. We denote by $\mathbb{Z}_2[z]$ the ring of polynomials with coefficients in $\mathbb{Z}_2$, by $\mathbb{L}_2$ the set of formal series of the form $\sum_{\ell=\ell_0}^{\infty} x_\ell z^{-\ell}$ with each $x_\ell \in \mathbb{Z}_2$ and $\ell_0 \in \mathbb{Z}$, and for any given integer $w \geq k$, we define $\varphi_w : \mathbb{L}_2 \to \mathbb{R}$ by

$$\varphi_w \left(\sum_{\ell=\ell_0}^{\infty} x_\ell z^{-\ell} \right) = \sum_{\ell=\max(\ell_0,1)}^{w} x_\ell 2^{-\ell}. \tag{5}$$

We select a *polynomial modulus* $Q = Q(z) \in \mathbb{Z}_2[z]$ of degree k, and a *generating vector* $\mathbf{a}(z) = (a_1(z), \ldots, a_s(z)) \in \mathbb{Z}_2[z]^s$, whose coordinates are polynomials of degrees less than k having no common factor with $Q(z)$. The point set of cardinality $n = 2^k$ is

$$P_n = \left\{ \left(\varphi_w \left(\frac{h(z) a_1(z)}{Q(z)} \right), \ldots, \varphi_w \left(\frac{h(z) a_s(z)}{Q(z)} \right) \right) : h(z) \in \mathbb{Z}_2[z], \deg(h(z)) < k \right\}. \tag{6}$$

Here, we want to optimize the vector $\mathbf{a}(z)$. This point set turns out to be a digital net in base 2 whose generating matrices $\mathbf{C}_j$ contain the first w digits of the binary expansion of the $a_j(z)/Q(z)$. These are Hankel matrices: each row is the previous one shifted to the left by one position, with the last entry determined by the recurrence with characteristic polynomial $Q(z)$, applied to the entries of the previous row. In theory, they have an infinite number of rows, but in practice we truncate them to $w \geq k$ rows. This finite w should be as large as possible to obtain a good approximation of the true PLR points. Typically, $w = 31$, but it could be $w = 63$ if we use 64-bit integers. See [8, 26, 36, 40, 39] for further details on PLRs.

A *high-order polynomial lattice rule* (HOPLR) of order α with $n = 2^k$ points is obtained by constructing an ordinary PLR with polynomial modulus $\tilde{Q}(z)$ of degree αk having $2^{\alpha k}$ points in s dimensions, and using only the first $n = 2^k$ points. See [1, 2, 7]. This type of construction can achieve a higher order of convergence for the error (almost $\mathcal{O}(n^{-\alpha})$) than an ordinary PLR for integrands f in a Sobolev space

of smoothness order α (i.e., when all mixed partial derivatives of up to order α are square integrable). One drawback is that because of the high degree of $\tilde{Q}$, the cost of a full CBC construction (see Section 5) is much higher since there are $2^{\alpha k}$ possibilities to examine each time we select a new coordinate of the generating vector.

Dick [3, 4] also proposed an *interlacing construction*, for digital nets in general (which includes PLRs), that can provide the higher-order convergence rate of almost $\mathcal{O}(n^{-\alpha})$ for the integration error, for integrands with smoothness order α. For an interlacing factor $d \in \mathbb{N}$, the method first constructs a digital net in sd dimensions, with generating matrices $C_1, \ldots, C_{sd}$. Then the generating matrices of the s-dimensional interlaced net are $C_1^{(d)}, \ldots, C_s^{(d)}$, where the rows of $C_j^{(d)}$ are the first rows of $C_{(j-1)d+1}, \ldots, C_{jd}$ in this order, then the second rows of these matrices in the same order, and so on.

The simplest way to define a RQMC point set from a digital net in base 2 is to add a digital random shift modulo 2 to all the points. To do this, we generate a single point $\mathbf{U} = (U_1, \ldots, U_s)$ uniformly in $(0,1)^s$, and perform a bitwise exclusive-or (XOR) between the binary digits of $\mathbf{U}$ and the corresponding digits of each point $\mathbf{u}_i$.

A more involved randomization method for digital nets is the *nested uniform scramble* (NUS) of Owen [45, 46]. In base 2, for each coordinate, we do the following. With probability 1/2, flip the first bit of all the points. Then, for the points whose first bit is 1, with probability 1/2, flip all the second bits. Do the same for the points whose first bit is 0, independently. Then do this recursively for all the bits. After all flipping is done for the first ℓ bits, partition the points in 2^ℓ batches according to the values of their first ℓ bits, and for each batch, flip bit $\ell+1$ of all the points with probability 1/2, independently across the batches. This requires $(2^\ell - 1)s$ random bits to flip the first ℓ bits of all coordinates. One can equivalently do this only for the first k bits, and generate the other bits randomly and independently across points [38].

A less expensive scramble, which gives less independence than NUS but more than a digital random shift, is a (left) *linear matrix scramble* (LMS) followed by a digital random shift (LMS+shift) [23, 24, 38, 48]. The LMS replaces $\mathbf{C}_j$ by $\tilde{\mathbf{C}}_j = \mathbf{L}_j \mathbf{C}_j \bmod 2$, where $\mathbf{L}_j$ is a random non-singular lower-triangular $w \times w$ binary matrix.

Owen [46] proved that under sufficient smoothness conditions on f, the RQMC variance with NUS on digital nets with fixed s and bounded t converges as $\mathcal{O}(n^{-3}(\log n)^{s-1})$. A variance bound of the same order was shown for LMS+shift in [23, 54]. Note that these results were proved under the assumption that $w = \infty$.

3 Hilbert Spaces and Projection-Dependent Weights

The FOMs used by LatNet Builder are based on generalized (weighted) Koksma-Hlawka inequalities of the form (3) where

$$\mathcal{V}^p(f) = \sum_{\emptyset \neq \mathfrak{u} \subseteq \{1,2,\ldots,s\}} \gamma_{\mathfrak{u}}^{-p} \mathcal{V}^p(f_{\mathfrak{u}}) \tag{7}$$

and

$$\mathscr{D}^q(P_n) = \sum_{\emptyset \neq \mathfrak{u} \subseteq \{1,2,\ldots,s\}} \gamma_{\mathfrak{u}}^q \mathscr{D}_{\mathfrak{u}}^q(P_n), \tag{8}$$

where $1/p + 1/q = 1$, $\gamma_{\mathfrak{u}} \in \mathbb{R}$ is a weight assigned to the subset $\mathfrak{u}$, $\mathscr{V}(f_{\mathfrak{u}})$ is the variation of $f_{\mathfrak{u}}$, $\mathscr{D}_{\mathfrak{u}}(P_n)$ is the discrepancy of the projection of P_n over the subset $\mathfrak{u}$ of coordinates, and $f = \sum_{\mathfrak{u} \subseteq \{1,2,\ldots,s\}} f_{\mathfrak{u}}$ is the functional ANOVA decomposition of f [10, 47]. LatNet Builder allows any $q \in [1,\infty]$. Taking $q = \infty$ with $p = 1$ means removing the q and taking the max instead of the sum in (8), while $p = \infty$ with $q = 1$ means removing the p and taking the max instead of the sum in (7). The most common choice is $p = q = 2$.

LatNet Builder implements a variety of choices for $\mathscr{D}_{\mathfrak{u}}(P_n)$, depending on the point set constructions. Some of these measures correspond to the worst-case error in some function space, assuming that the points of P_n are not randomized. Others correspond to the mean-square error (or variance), assuming that the points are randomized in some particular way. This is typically done by defining a RKHS with a kernel that is invariant with respect to the given randomization (i.e., digital shift-invariant, scramble-invariant, etc.), and taking the worst-case error in that space.

The role of the weights is to better recognize the importance of the subsets $\mathfrak{u}$ for which $f_{\mathfrak{u}}$ contributes the most to the error or variance. That is, if $\mathscr{V}(f_{\mathfrak{u}})$ is unusually large, we want to divide it by a larger weight $\gamma_{\mathfrak{u}}$ to control its contribution to $\mathscr{V}(f)$, but then we have to multiply $\mathscr{D}_{\mathfrak{u}}(P_n)$ in (8) by the same weight. The final effect is that the FOM will penalize more the discrepancy for that particular projection.

In principle, the weights $\gamma_{\mathfrak{u}}$ can be arbitrary. But for large s, defining arbitrary individual weights for the $2^s - 1$ projections is impractical, so special forms of weights that are parameterized by much fewer than $2^s - 1$ parameters have been proposed. The most common ones are *product weights*, for which a weight γ_j is assigned to coordinate j for $j = 1,\ldots,s$, and $\gamma_{\mathfrak{u}} = \prod_{j \in \mathfrak{u}} \gamma_j$; *order-dependent weights*, for which $\gamma_{\mathfrak{u}} = \Gamma_{|\mathfrak{u}|}$ where $\Gamma_1,\ldots,\Gamma_s$ are selected constants and $|\mathfrak{u}|$ is the cardinality of of $\mathfrak{u}$; and the *product-and-order-dependent (POD) weights*, which are a combination of the two, with $\gamma_{\mathfrak{u}} = \Gamma_{|\mathfrak{u}|} \prod_{j \in \mathfrak{u}} \gamma_j$. These are all available in LatNet Builder. For more discussion on how to select the weights, see [8, 12, 32, 33, 34], for example.

LatNet Builder can construct point sets that are extensible in the number of dimensions and also in the number of points, which means that we can construct point sets that perform well in the first s dimensions for $s = s_{\min},\ldots,s_{\max}$, and/or if we take the first n points for $n = n_1, n_2, \ldots, n_m$, simultaneously. Typically, one would take $n_j = 2^{k_{\min}+j-1}$ for $j = 1,\ldots,m$, so $n_m = 2^{k_{\max}} = 2^{k_{\min}+m-1}$ [22]. The global FOM in this case will be a weighted sum or maximum of the FOMs over the considered dimensions s and/or cardinalities n_j. The CBC construction approach described in Section 5 already gives a way to implement the extension in s. For the extension in n (or k), LatNet Builder implements criteria and heuristic search methods that account for a global FOM.

4 Figures of Merit

In this section, we review the FOMs implemented in LatNet Builder. Most of them have the general form (8) where typically, when the points have the appropriate special structure of a lattice, polynomial lattice, or digital net, and with an adapted FOM, we have

$$\mathscr{D}^q_{\mathfrak{u}}(P_n) = \frac{1}{n}\sum_{i=0}^{n-1}\prod_{j\in\mathfrak{u}}\phi(u_{i,j}) \tag{9}$$

for some function $\phi : [0,1) \to \mathbb{R}$. With product weights $\gamma_{\mathfrak{u}} = \prod_{j\in\mathfrak{u}}\gamma_j$, this becomes

$$\mathscr{D}^q(P_n) = -1 + \frac{1}{n}\sum_{i=0}^{n-1}\prod_{j=1}^{s}(1+\gamma_j^q\phi(u_{i,j})),$$

which can be computed with $\mathscr{O}(ns)$ evaluations of ϕ.

As an illustration, for a randomly-shifted lattice rule, the variance is:

$$\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}] = \sum_{\mathbf{0}\neq\mathbf{h}\in L_s^*}|\hat{f}(\mathbf{h})|^2, \tag{10}$$

where $L_s^* \subset \mathbb{Z}^s$ is the *dual lattice* [30]. It is also known that for periodic continuous functions having square-integrable mixed partial derivatives up to order $\alpha/2$ for an even integer $\alpha \geq 2$, one has $|\hat{f}(\mathbf{h})|^2 = \mathscr{O}((\max(1,h_1)\cdots\max(1,h_s))^{-\alpha})$. This motivates the well-known FOM [33, 40, 50]:

$$\begin{aligned}\mathscr{P}_\alpha &:= \sum_{\mathbf{0}\neq\mathbf{h}\in L_s^*}(\max(1,h_1)\cdots\max(1,h_s))^{-\alpha} \\ &= \frac{1}{n}\sum_{i=0}^{n-1}\sum_{\emptyset\neq\mathfrak{u}\subseteq\{1,\ldots,s\}}\left(\frac{-(-4\pi^2)^{\alpha/2}}{\alpha!}\right)^{|\mathfrak{u}|}\prod_{j\in\mathfrak{u}}B_\alpha(u_{i,j})\end{aligned} \tag{11}$$

where $B_{\alpha/2}$ is the Bernoulli polynomial of degree $\alpha/2$ ($B_1(u) = u - 1/2$, $B_2(u) = u^2 - u + 1/6$, etc.), and the equality in (11) holds only when α is an even integer. Moreover, there are rank-1 lattices point sets P_n for which $\mathscr{P}_\alpha$ converges as $\mathscr{O}(n^{-\alpha+\varepsilon})$ for any $\varepsilon > 0$ [9, 49, 50]. Adding projection-dependent weights $\gamma_{\mathfrak{u}}$ leads to the weighted $\mathscr{P}_{\gamma,\alpha}$, defined by (8) and (9) with $q = 2$,

$$\phi(u_{i,j}) = -(-4\pi^2)^{\alpha/2}B_\alpha(u_{i,j})/\alpha!,$$

and $\mathscr{D}^2_{\mathfrak{u}}(P_n) = \mathscr{P}_{\alpha,\mathfrak{u}}(P_n)$ is the $\mathscr{P}_\alpha$ for the projection of P_n on the coordinates in $\mathfrak{u}$.

There is a similar variance expression for digital nets in base 2 with a random digital shift, with the Fourier coefficients $\hat{f}(\mathbf{h})$ replaced the the Walsh coefficients, and the dual lattice replaced by the dual net [8, Definition 4.76] or the dual lattice in the case of PLRs [26, 36]. Thus, FOMs that correspond to variance bounds can be obtained by finding easily computable bounds on the Walsh coefficients. By assuming a rate of decrease of $\mathscr{O}(2^{-\alpha|\mathbf{h}|})$ of the Walsh coefficients $\tilde{f}(\mathbf{h})$ with

$\mathbf{h} = (h_1, \ldots, h_s) \in \mathbb{N}^s$ and $|\mathbf{h}| = |h_1| + \cdots + |h_s|$, and using a RKHS with shift- and scramble-invariant kernel, [54] and [6] obtain a FOM of the form (8) and (9) with

$$\phi(x) = \phi_\alpha(x) = \mu(\alpha) - \mathbb{I}[x > 0] \cdot 2^{(1+\lfloor \log_2(x) \rfloor)(\alpha-1)}(\mu(\alpha)+1),$$

where $\mathbb{I}$ is the indicator function, $-\lfloor \log_2 x \rfloor$ is the index of the first nonzero digit in the expansion of x, and $\mu(\alpha) = \left(1 - 2^{1-\alpha}\right)^{-1}$ for any real number $\alpha > 1$. This gives $\mu(2) = 2$, $\mu(3) = 4/3, \ldots,$ For $\alpha = 2$, this gives

$$\phi_2(x) = 2(1 - \mathbb{I}[x > 0] \cdot 3 \cdot 2^{\lfloor \log_2(x) \rfloor}),$$

which corresponds to the FOM suggested in [36, Section 6.3] for PLRs. In [23, 54], $\phi(x)$ is written in terms of $\eta = \alpha - 1$ instead, but it is exactly equivalent. These papers also show the existence of digital nets for which the FOM converges as $\mathcal{O}(n^{-\alpha}(\log n)^{s-1})$ for any $\alpha > 1$. This FOM can be seen as a counterpart of $\mathscr{P}_\alpha$ and we call it $\tilde{\mathscr{P}}_\alpha$. Its value $\tilde{\mathscr{P}}_{\alpha,\mathfrak{u}}(P_n)$ on the projection of P_n on the coordinates in $\mathfrak{u}$ can be used for $\mathscr{D}^2_\mathfrak{u}(P_n)$, with $q = 2$. Note that under our assumption that the first k rows of each generating matrix are linearly independent, $-\lfloor \log_2(u_{i,j}) \rfloor$ never exceeds k when $u_{i,j} \neq 0$, and therefore this FOM depends only on the first k bits of output.

Dick and Pillichshammer [8, Chapter 12] consider a RKHS with shift-invariant kernel, which is a weighted Sobolev space of functions whose mixed partial derivatives of order 1 are square-integrable. This gives a FOM of the form (8) and (9) with $q = 2$ and

$$\phi(x) = 1/6 - \mathbb{I}[x > 0] \cdot 2^{\lfloor \log_2(x) \rfloor - 1}.$$

They show that there are digital nets for which this FOM (and therefore the square error) converges almost as $\mathcal{O}(n^{-2})$. In their Chapter 13, they find that the scramble-invariant version gives the same ϕ. Note that this $\phi(x)$ is equal to $\phi_2(x)$ above, divided by 12. Therefore, we can get the corresponding FOM just from $\tilde{\mathscr{P}}_2$ by multiplying the weights by order-dependent factors of $1/12^j$ for order j.

Goda [13] examines *interlaced polynomial lattice rules* (IPLR), also for a Sobolev space of order α, with an interlacing factor $d > 1$. He provides two upper bounds on the worst-case error in a deterministic setting. These bounds can be used as FOMs. The first is valid for all positive integer values of α and $d > 1$, whereas the second holds only for $1 < d \leq \alpha$, but is tighter when it applies. These two bounds have the form (8) and (9) with $q = 1$, $\gamma_\mathfrak{u}$ replaced by $\tilde{\gamma}_\mathfrak{u}$, and

$$\phi(x_{i,j}) = -1 + \prod_{\ell=1}^{d}(1 + \phi_{\alpha,d,\ell}(x_{i,(j-1)d+\ell})),$$

where for the first bound, $\tilde{\gamma}_\mathfrak{u} = \gamma_\mathfrak{u} 2^{\alpha(2d-1)|\mathfrak{u}|/2}$,

$$\phi_{\alpha,d,\ell}(x) = \frac{1 - 2^{(\min(\alpha,d)-1)\lfloor \log_2 x \rfloor}(2^{\min(\alpha,d)} - 1)}{2^{(\alpha+2)/2}(2^{\min(\alpha,d)-1} - 1)}$$

for all $x \in [0,1)$, where $2^{\lfloor \log_2 0 \rfloor} = 0$, while for the second bound, $\tilde{\gamma}_\mathfrak{u} = \gamma_\mathfrak{u}$ and

$$\phi_{\alpha,d,\ell}(x) = \frac{2^{d-1}(1-2^{(d-1)\lfloor \log_2 x\rfloor}(2^d-1))}{2^\ell(2^{d-1}-1)}.$$

One can achieve a convergence rate of almost $\mathcal{O}(n^{-\min(\alpha,d)})$ for these FOMs (and therefore for the worst-case error). We denote these two FOMs by $\mathscr{I}^{(a)}_{\alpha,d}$ and $\mathscr{I}^{(b)}_{\alpha,d}$.

Goda and Dick [14] proposed another FOM, also for a Sobolev space of order α, for interlaced randomly-scrambled PLRs of high order. They showed that this scheme can achieve the best possible convergence rate of $\mathcal{O}(n^{-(2\min(\alpha,d)+1)+\delta})$ for the variance. The FOM, denoted $\mathscr{I}^{(c)}_{\alpha,d}$, has the same form, but with $q=2$,

$$\phi(x) = \phi_{\alpha,d}(x) = \frac{1-2^{2\min(\alpha,d)\lfloor \log_2 x\rfloor}(2^{2\min(\alpha,d)+1}-1)}{2^\alpha(2^{2\min(\alpha,d)}-1)},$$

and $\gamma_\mathfrak{u}$ replaced by $\tilde{\gamma}_\mathfrak{u} = \gamma_\mathfrak{u} D_{\alpha,d}^{|\mathfrak{u}|}$ where $D_{\alpha,d} = 2^{2\max(d-\alpha,0)+(2d-1)\alpha}$.

Another set of FOMs are obtained from upper bounds on the star discrepancy of $\mathscr{D}^*(P_n)$ or its projections on subsets of coordinates, when P_n is a digital (t,k,s)-net. One such bound is $\mathscr{D}^*(P_n) \le 1-(1-1/n)^s + \mathscr{R}_2$ where

$$\mathscr{R}_2 = -1 + \frac{1}{n}\sum_{i=0}^{n-1}\prod_{j=1}^{s}\left[\sum_{k=0}^{n-1} 2^{-\lfloor \log_2 k\rfloor - 1}\mathrm{wal}_k(u_{i,j})\right] \tag{12}$$

wal_k is the kth Walsh function in one dimension, and we assume that the generating matrices $\mathbf{C}_j$ are $k\times k$. See [8, Theorems 5.34 and 5.36], where a more general version with projection-dependent weights is also given. For PLRs in base $b=2$, this criterion is equal to $\mathscr{R}'_{2,\gamma}$ given in [8, Chapter 10]:

$$\mathscr{R}'_{2,\gamma} = -\sum_{\emptyset\ne\mathfrak{u}\subseteq\{1,2,\ldots,s\}}\gamma_\mathfrak{u} + \frac{1}{n}\sum_{i=0}^{n-1}\sum_{\emptyset\ne\mathfrak{u}\subseteq\{1,2,\ldots,s\}}\gamma_\mathfrak{u}\prod_{j\in\mathfrak{u}}\phi_k(u_{i,j}) \tag{13}$$

where $\phi_k(u) = -\lfloor \log_2(u)\rfloor/2$ if $u \ge 2^{-k}$ and $\phi_k(u) = 1+k/2$ otherwise.

A classical upper bound on the star discrepancy is also given by the t-value of the digital net:

$$\mathscr{D}^*(P_n) \le \frac{2^t}{n}\sum_{j=0}^{s-1}\binom{k-t}{j}.$$

If we use this upper bound for each projection on the subset $\mathfrak{u}$ of coordinates, we get the FOM (8) with $q=1$ and

$$\mathscr{D}_\mathfrak{u}(P_n) = \frac{2^{t_\mathfrak{u}}}{n}\sum_{j=0}^{|\mathfrak{u}|-1}\binom{k-t_\mathfrak{u}}{j}$$

where $t_\mathfrak{u} = t_\mathfrak{u}(P_n)$ is the t-value of the projection of P_n on the coordinates in $\mathfrak{u}$. LatNet Builder implements this with arbitrary weights, using algorithms described in [37].

Dick [4] obtains worst-case error bounds that converge at rate almost $\mathcal{O}(n^{-\alpha})$ for interlaced digital nets, based on the t-values of the projections.

5 Search Methods

For given construction type, FOM, and weights, finding the best choice of parameters may require to try all possibilities, but their number is usually much too large. LatNet Builder implements the following search methods.

In an *exhaustive search*, all choices of parameters are tried, so we are guaranteed to find the best one. This is possible only when there are not too many possibilities.

A *random search* samples uniformly and independently a fixed number r of parameter choices, and the best one is retained.

In a *full component-by-component (CBC)* construction, the parameters are selected for one coordinate at a time, by taking into account the choices for the previous coordinates only [5, 8, 51]. The parameters for coordinate j (e.g., the jth coordinate of the generating vector in the case of lattices), are selected by minimizing the FOM for the first j coordinates, in j dimensions, by examining all possibilities of parameters for this coordinate, without changing the parameter choices for the previous coordinates. This is done for the s coordinates in succession. This greedy approach can reduce by a huge factor (exponential in the dimension) the total number of cases that are examined in comparison with the exhaustive search. What is very interesting is that for most types of QMC constructions and FOMs implemented in LatNet Builder, the convergence rate for the worst-case error or variance obtained with this restricted approach is provably the same as for the exhaustive search [8, 15].

For lattice-type point sets, with certain FOMs and choices of weights (e.g., $\mathscr{P}_2$ and $\tilde{\mathscr{P}}_2$ with product and/or order-dependent weights), a *fast CBC* construction can be implemented by using a fast Fourier transform (FFT), so the full CBC construction can be performed much faster [8, 44, 42]. LatNet Builder supports this.

When the number of choices for each coordinate is too large or fast-CBC does not apply, one can examine only a fixed number of random choices for each coordinate j; we call this the *random CBC* construction.

For lattice-type constructions, one can also further restrict the search to *Korobov*-type generating vectors. The first coordinate is set to 1 and only the second coordinate needs to be selected. This can be done either by an exhaustive search or by just taking a random sample for the second coordinate (*random Korobov*).

For digital nets, a *mixed CBC* method is also available: it uses full CBC for the first $d-1$ coordinates and random CBC for the other ones, for given $1 \leq d \leq s$.

6 Usage of the Software Tool

At the first level, LatNet Builder is a library written in C++ which implements classes and methods to compute FOMs and search for good point sets for all the construction methods and FOMs discussed in this paper. The source code and a detailed reference manual for the library are provided at https://github.com/umontreal-simul/latnetbuilder. The library can be used directly from C++ programs and can be extended if desired. This is the most flexible option, but it requires knowledge of C++ and the library.

At the second level, there is an executable `latnetbuilder` program that can be called directly from the command line in Linux, Mac OS, or Windows. This program has a large number of options to specify the type and number of points, number of dimensions, search method, FOM, weights, output file format, etc. We think this is the most convenient way of using LatNet Builder in practice. In addition to giving the search results on the terminal, the program creates a directory with two output files: one summarizes the search parameters and the other one puts the parameters of the selected point set in a standard format designed for reading by other software that can generate and use the RQMC points in applications. There are selected file formats for ordinary lattice rules, polynomial lattice rules, Sobol points, and general digital nets in base 2. A tutorial on the command line and a summary of the options can be found in the reference manual.

At a third level, there is a Java interface in SSJ, a Python interface included in the distribution, and a Graphical User Interface (GUI) based on the Jupyter ecosystem, written in Python. These interfaces use the command line internally. With the GUI, the user can select the desired options in menus, write numerical values in input cells, write the name of the desired output directory, and launch a search. The LatNet Builder program with the Python interface and the GUI can be installed as a Docker container on one's machine. An even simpler access to the GUI is available without installing anything: just click on the "Launch Binder" black and pink link on the GitHub site and it will run the GUI with a version of the program hosted by Binder. This service provides limited computation resources but is convenient for small experiments and to get a sense of what the software does.

We now give examples to illustrate how the command line works, how the results look like, and give some idea of the required CPU times for the search. The timings were made in a VirtualBox for Ubuntu Linux running atop Windows 10 on an old desktop computer with an Intel i7-2600 processor at 3.4GHz and 32 Gb of memory.

The following command makes a search for a polynomial lattice rule with $n = 2^{16}$ points in 256 dimensions, with the default irreducible polynomial modulus, using the fast-CBC search method, the $\tilde{P}_2$ criterion, $q = 2$, and order-dependent weights $\Gamma_2 = 10$, $\Gamma_3 = 0.1$, $\Gamma_4 = 0.001$, and the other weights equal to 0 (recall that Γ_1 has no impact on the selection). These weights decrease quickly with the order because the number of projections of any given order increases very quickly with the order. If they decrease too slowly, the total weight of the projections of order 2 in the FOM will be negligible compared to those of order 4, for instance. For a smaller s, the weights may decrease more slowly.

```
latnetbuilder -t lattice -c polynomial -s 2^16 -d 256 -e fast-CBC
    -f CU:P2 -q 2 -w order-dependent:0,0,10.,0.1,0.001 -O lattice
```

This search takes about 840 seconds to complete and the retained rule had an FOM of 60.235. About 70% of this FOM is contributed by the projections of order 4 and less than 1% by the projections of order 2. The output file looks like:

```
# Parameters for a polynomial lattice rule in base 2
256     # s = 256 dimensions
16      # n = 2^16 = 65536 points
96129   # polynomial modulus
1       # coordinates of generating vector, starting at j=1
47856
60210
44979
  ⋮
```

Instead of making the search directly in the polynomial lattice space, we can make the same search by viewing the PLR as a digital net, using the option "`-t net`." With that option, fast-CBC search is not available, but we can do a random CBC search, say with 100 samples for each coordinate. With either search method, instead of reporting the modulus and generating vector of the PLR in the output file, we can have all the columns of the generating matrices, by using the option "`-O net`" instead of "`-O lattice`." The command is:

```
latnetbuilder -t net -c polynomial -s 2^16 -d 256
        -e random-CBC:100 -f CU:P2 -q 2
        -w order-dependent:0,0,10.,0.1,0.001 -O net
```

With this, the search takes about 160 seconds and gives an FOM of 63.25. The output file looks like this, with 16 integers per dimension, one integer for each column:

```
# Parameters for a digital net in base 2
256  # s = 256 dimensions
16   # n = 2^16 = 65536 points
31   # r = 31 binary output digits
# Columns of gen. matrices C_1,...,C_s, one matrix per line:
33260 66520 133040 266081 532162 1064325 2128651 4257303  ...
1561357389 975231131 1950462262 1753440876 1359398105 ...
1642040599 1136597551 125711455 251422911 502845823 ...
  ⋮
```

For a search in 32 dimensions instead of 256, it takes about 40 seconds for the first case and 8 seconds for the second case.

As another example, the following command launches a search for good direction numbers for Sobol' points for up to 2^{16} points in 256 dimensions. It uses a mixed CBC search which does a full CBC evaluation for the first 10 coordinates and then a random-CBC search with 100 random samples for each of the remaining coordinates. The criterion is the maximum t-value over projections of order 2 and 3, with order-dependent weights of $\Gamma_2 = 1.0$, $\Gamma_3 = 0.5$, and 0 for everything else. Here, since we take the sup over the projections, the weights can decrease much more slowly.

```
latnetbuilder -t net -c sobol -s 2^16 -d 256 -e mixed-CBC:100:10
     -f projdep:t-value -q inf -w order-dependent:0:0,1.0,0.5
```

The search took about 3340 seconds and returned a FOM of 8.0. The output file provides the retained direction numbers and it looks like this:

```
# Initial direction numbers m_{j,c} for Sobol points
# s = 256 dimensions
1    # This is m_{j,k} for the second coordinate
1 1
1 1 1
1 1 1
1 1 5 1
1 3 5 1
1 1 1 9 9
1 1 1 9 17
  ⋮
```

If we change s to 32, the search takes 12 seconds and the FOM is 5.0 instead.

7 Simple Numerical Illustrations

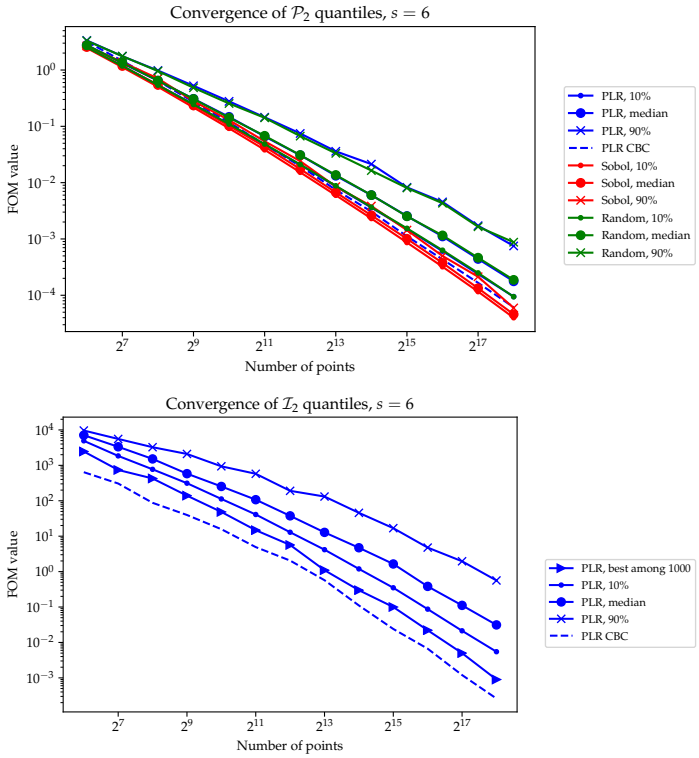
Here we give a few simple examples of what LatNet Builder can do. The simulation experiments, including the generation and randomization of the points, were done using SSJ [29].

7.1 FOM quantiles for different constructions

One might be interested in estimating the probability distribution of FOM values obtained when selecting parameters at random for a given type of construction, perhaps under some constraints, and as a function of n. Here we estimate this distribution by its empirical counterpart with an independent sample of size 1000 (with replacement), and we report the 0.1, 0.5 and 0.9 quantiles of this empirical distribution, for n going from 2^6 to 2^{18}. We do this for PLRs, Sobol' points, and digital nets with arbitrary invertible and projection-regular generating matrices (random nets), with $\tilde{\mathscr{P}}_2$ taken as the FOM, in $s = 6$ dimensions, with $\gamma_{\mathfrak{u}} = 0.7^{|\mathfrak{u}|}$ for all $\mathfrak{u}$. We also report the value obtained by a (full) fast CBC search for a PLR. The results are displayed in the first panel of Figure 1. We see that the FOM distribution has a smaller mean and much less variance for the Sobol' points than for the other contructions. Even the median obtained for Sobol' beats (slightly) the FOM obtained by a full CBC construction with PLRs. The quantiles for random PLRs and random nets are approximately the same.

The second panel of the figure shows the results of a similar sampling for PLRs with $\mathscr{S}_{2,2}^{(c)}$ as a FOM, also in 6 dimensions. Here, the FOM values are more dispersed and the fast CBC gives a significantly better value than the best FOM obtained by random sampling. Also the search for the point set parameters is much quicker with the fast CBC construction than with random sampling of size 1000.

Fig. 1: The 0.1, 0.5, and 0.9 quantiles of the FOM distribution as functions of n for various constructions, in log-log scale.

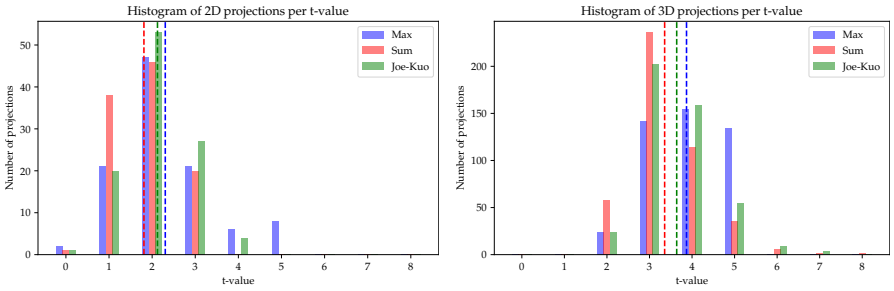


7.2 Comparison with tabulated parameter selections

We now give small examples showing how searching for custom parameter values with LatNet Builder can make a difference in the RQMC variance compared with pre-tabulated parameter values available in software or over the Internet. We do this for Sobol' nets, and our comparison is with the precomputed direction numbers obtained by Joe and Kuo [25], which are arguably the best proposed values so far. These parameters were obtained by optimizing a FOM based on the t-values over two-dimensional projections, using a CBC construction. With LatNet Builder, we can account for any selected projections in our FOM. For instance, if we think all the projections in two and three dimensions are important, we can select a FOM that accounts for all these projections. To illustrate this, we made a CBC construction of $n = 2^{12}$ Sobol' points in $s = 15$ dimensions, using the sum or the maximum of t-values in the two- and three-dimensional projections. Figure 2 shows the distribution of t-values obtained with the sum, the max, and the points from [25]. Compared with the latter, we are able to reduce the worse t-value over 3-dim projections from 8 to 5

when using the max, and to reduce the average t-value when using the sum. However, when using the max, we get a few poor two-dim projections, because we compare the t-values on the same scale for two and three dimensions. We should probably multiply the t-value by a scaling factor that decreases with the dimension.

Fig. 2: Distributions of t-values for 2-dim and 3-dim projections, for three Sobol' point sets: (1) *Joe-Kuo* taken from [25], (2) *Max* and (3) *Sum* are found by LatNet Builder as explained in the text. For each case, we report the number of projections having any given t-value, as well as the average t-value (dashed vertical lines).



In our next illustration, we compare the RQMC variances for Sobol' points with direction numbers taken from [25] and direction numbers found by LatNet Builder using a custom FOM for our function. We want to integrate

$$f(\mathbf{u}) = \prod_{j=1}^{5}(\psi(u_j)-\mu) + \prod_{j=6}^{10}(\psi(u_j)-\mu),$$

where $\psi(u) = \big((u-0.5)^2+0.05\big)^{-1}$ and $\mu = \mathbb{E}[\psi(U)] \approx 10.3$ when $U \sim U(0,1)$. This function is the sum of two five-dimensional ANOVA terms for a more general function taken from [11]. A good FOM for this function should focus mainly on these two five-dim projections, namely $\mathfrak{u} = \{1,2,3,4,5\}$ and $\mathfrak{u} = \{6,7,8,9,10\}$, and not on the two-dim projections as in [25]. So we made a search with the $\tilde{\mathscr{P}}_2$ criterion with weights $\gamma_{\mathfrak{u}} = 1$ for these two projections and 0 elsewhere, to obtain new direction numbers for $n = 2^{20}$ Sobol' points in 10 dimensions. Then we estimated the variance of the sample RQMC average over these n points with the two choices of direction numbers (those of [25] and ours), using $m = 200$ independent replications of an RQMC scheme that used only a random digital shift. The empirical variance with our custom points was smaller by a factor of more than 18.

7.3 Variance for another toy function

Here we consider a family of test functions similar to those in [53]:

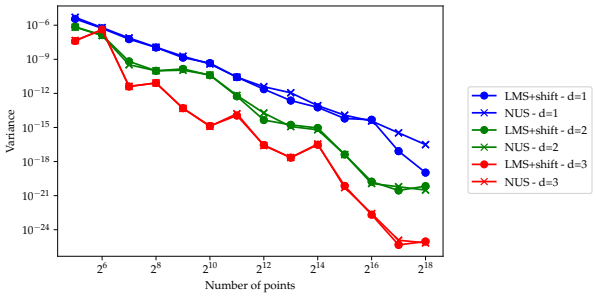
$$f_{s,\mathbf{c}}(\mathbf{u}) = \prod_{j=1}^{s} (1 + c_j \cdot (u_j - 1/2))$$

for $\mathbf{u} \in (0,1)^s$, where $\mathbf{c} = (c_1, \ldots, c_s) \in (0,1)^s$. The ANOVA components are, for all $\mathfrak{u} \subset \{1, \ldots, s\}$,

$$(f_{s,\mathbf{c}})_{\mathfrak{u}}(\mathbf{u}) = \prod_{j \in \mathfrak{u}} c_j \cdot (u_j - 1/2)).$$

For an experiment, we take arbitrarly $s = 3$ and $\mathbf{c} = (0.7, 0.2, 0.5)$. We use LatNet Builder to search for good PLRs with a fast CBC construction, with product weights $\gamma_j = c_j$, with the FOMs $\mathscr{P}_2$, $\mathscr{I}^{(c)}_{2,2}$, and $\mathscr{I}^{(c)}_{3,3}$ (whose interlacing factors d are 1, 2, and 3, respectively). For each $n = 2^k$, $k = 5, \ldots, 18$, we estimate the RQMC variance with m independent replications of the randomization scheme, with $m = 1000$ for LMS+shift, and $m = 100$ for NUS. For the interlaced points, the randomization is performed before the interlacing, as in [14]. Figure 3 shows the variance as a function of n, in log-log scale. We see that the two randomization schemes give approximately the same variance. However, the time to generate and randomize the points is much larger for NUS than for LMS+shift: around 10 times longer for 2^{11} points and 50 times longer for 2^{18} points. As expected, the variance reduction and the convergence rate are larger when the interlacing factor increases, although the curves are more noisy.

Fig. 3: Variance as a function of n in log-log scale, for PLRs with two randomization schemes and three interlacing factors d, found with LatNet Builder. We also report the average time to generate and randomize the points with LMS+shift.



Nb. of points	Constr. time
128	0.02 s
512	0.04 s
2048	0.12 s
8192	0.50 s
32768	2.25 s
131072	8.32 s

8 Conclusion

LatNet Builder is both a tool for researchers to study the properties of highly uniform point sets and associated figures of merit, and for practitioners who want to find good

parameters for a specific task. It is relatively easy to incorporate new FOMs into the software, especially if they are in the kernel form (9).

Many questions remain open regarding the roles of the construction, the search method, the randomization, and (perhaps more importantly) the choice of the weights. It is our hope that the software presented here will spur interest into these issues.

Acknowledgements This work has been supported by a NSERC Discovery Grant and an IVADO Grant to P. L'Ecuyer, and by a stipend from Corps des Mines to P. Marion. F. Puchhammer was supported by Spanish and Basque governments fundings through BCAM (ERDF, ESF, SEV-2017-0718, PID2019-108111RB-I00, PID2019-104927GB-C22, BERC 2018e2021, EXP. 2019/00432, KK-2020/00049), and the computing infrastructure of i2BASQUE and IZO-SGI SGIker (UPV). Yocheved Darmon wrote code for producing the output files.

References

1. Baldeaux, J., Dick, J., Greslehner, J., Pillichshammer, F.: Construction algorithms for higher order polynomial lattice rules. Journal of Complexity **27**, 281–299 (2011)
2. Baldeaux, J., Dick, J., Leobacher, G., Nuyens, D., Pillichshammer, F.: Efficient calculation of the worst-case error and (fast) component-by-component construction of higher order polynomial lattice rules. Numerical Algorithms **59**(3), 403–431 (2012)
3. Dick, J.: Explicit constructions of quasi-Monte Carlo rules for the numerical integration of high-dimensional periodic functions. SIAM Journal on Numerical Analysis **45**(5), 2141–2176 (2007)
4. Dick, J.: Walsh spaces containing smooth functions and quasi-Monte Carlo rules of arbitrary high order. SIAM Journal on Numerical Analysis **46**(3), 1519–1553 (2008)
5. Dick, J., Kuo, F.Y., Pillichshammer, F., Sloan, I.H.: Construction algorithms for polynomial lattice rules for multivariate integration. Mathematics of Computation **74**(248), 1895–1921 (2005)
6. Dick, J., Pillichshammer, F.: Multivariate integration in weighted Hilbert spaces based on Walsh functions and weighted Sobolev spaces. Journal of Complexity **21**(2), 149–195 (2005)
7. Dick, J., Pillichshammer, F.: Strong tractability of multivariate integration of arbitrary high order using digitally shifted polynomial lattice rules. Journal of Complexity **23**, 436–453 (2007)
8. Dick, J., Pillichshammer, F.: Digital Nets and Sequences: Discrepancy Theory and Quasi-Monte Carlo Integration. Cambridge University Press, Cambridge, U.K. (2010)
9. Dick, J., Sloan, I.H., Wang, X., Woźniakowski, H.: Good lattice rules in weighted Korobov spaces with general weights. Numerische Mathematik **103**, 63–97 (2006)
10. Efron, B., Stein, C.: The jackknife estimator of variance. Annals of Statistics **9**, 586–596 (1981)
11. Genz, A.: Testing multidimensional integration routines. In: Proceedings of the International Conference on Tools, Methods and Languages for Scientific and Engineering Computation, pp. 81–94. Elsevier North-Holland (1984)
12. Gilbert, A., Kuo, F., Sloan, I.: Hiding the weights—CBC black box algorithms with a guaranteed error bound. Mathematics and Computers in Simulation **143**, 202–214 (2018)
13. Goda, T.: Good interlaced polynomial lattice rules for numerical integration in weighted Walsh spaces. Journal of Computational and Applied Mathematics **285**, 279–294 (2015)
14. Goda, T., Dick, J.: Construction of interlaced scrambled polynomial lattice rules of arbitrary high order. Foundation of Computational Mathematics **15**, 1245–1278 (2015)
15. Goda, T., Suzuki, K.: Recent advances in higher order quasi-Monte Carlo methods. In: Discrepancy Theory, pp. 69–102. De Gruyter (2019)

16. Goda, T., Suzuki, K., Yoshiki, T.: An explicit construction of optimal order quasi-Monte Carlo rules for smooth integrands. SIAM Journal on Numerical Analysis **54**, 2664—2683 (2016)
17. Goda, T., Suzuki, K., Yoshiki, T.: Optimal order quasi-Monte Carlo integration in weighted Sobolev spaces of arbitrary smoothness. IMA Journal on Numerical Analysis **37**, 505–518 (2017)
18. Goda, T., Suzuki, K., Yoshiki, T.: Optimal order quadrature error bounds for infinite-dimensional higher-order digital sequences. Foundation of Computational Mathematics **18**, 433–458 (2018)
19. Hickernell, F.J.: A generalized discrepancy and quadrature error bound. Mathematics of Computation **67**(221), 299–322 (1998)
20. Hickernell, F.J.: Lattice rules: How well do they measure up? In: P. Hellekalek, G. Larcher (eds.) Random and Quasi-Random Point Sets, *Lecture Notes in Statistics*, vol. 138, pp. 109–166. Springer-Verlag, New York (1998)
21. Hickernell, F.J.: Obtaining $O(N^{-2+\varepsilon})$ convergence for lattice quadrature rules. In: K.T. Fang, F.J. Hickernell, H. Niederreiter (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2000, pp. 274–289. Springer-Verlag, Berlin (2002)
22. Hickernell, F.J., Hong, H.S., L'Ecuyer, P., Lemieux, C.: Extensible lattice sequences for quasi-Monte Carlo quadrature. SIAM Journal on Scientific Computing **22**(3), 1117–1138 (2001)
23. Hickernell, F.J., Yue, R.X.: The mean square discrepancy of scrambled (t,s)-sequences. SIAM Journal on Numerical Analysis **38**(4), 1089–1112 (2001)
24. Hong, H.S., Hickernell, F.J.: Algorithm 823: Implementing scrambled digital sequences. ACM Transactions on Mathematical Software **29**, 95–109 (2003)
25. Joe, S., Kuo, F.Y.: Constructing Sobol sequences with better two-dimensional projections. SIAM Journal on Scientific Computing **30**(5), 2635–2654 (2008)
26. L'Ecuyer, P.: Polynomial integration lattices. In: H. Niederreiter (ed.) Monte Carlo and Quasi-Monte Carlo Methods 2002, pp. 73–98. Springer-Verlag, Berlin (2004)
27. L'Ecuyer, P.: Quasi-Monte Carlo methods with applications in finance. Finance and Stochastics **13**(3), 307–349 (2009)
28. L'Ecuyer, P.: SSJ: Stochastic simulation in Java (2016). http://simul.iro.umontreal.ca/ssj/, accessed 9th August 2021
29. L'Ecuyer, P., Buist, E.: Simulation in Java with SSJ. In: Proceedings of the 2005 Winter Simulation Conference, pp. 611–620. IEEE Press (2005)
30. L'Ecuyer, P., Lemieux, C.: Variance reduction via lattice rules. Management Science **46**(9), 1214–1235 (2000)
31. L'Ecuyer, P., Lemieux, C.: Recent advances in randomized quasi-Monte Carlo methods. In: M. Dror, P. L'Ecuyer, F. Szidarovszky (eds.) Modeling Uncertainty: An Examination of Stochastic Theory, Methods, and Applications, pp. 419–474. Kluwer Academic, Boston (2002)
32. L'Ecuyer, P., Munger, D.: Constructing adapted lattice rules using problem-dependent criteria. In: Proceedings of the 2012 Winter Simulation Conference, pp. 373–384. IEEE Press (2012)
33. L'Ecuyer, P., Munger, D.: On figures of merit for randomly-shifted lattice rules. In: H. Woźniakowski, L. Plaskota (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2010, pp. 133–159. Springer-Verlag, Berlin (2012)
34. L'Ecuyer, P., Munger, D.: Algorithm 958: Lattice builder: A general software tool for constructing rank-1 lattice rules. ACM Transactions on Mathematical Software **42**(2), Article 15 (2016)
35. Lemieux, C., Cieslak, M., Luttmer, K.: RandQMC User's Guide: A Package for Randomized Quasi-Monte Carlo Methods in C (2004). Software user's guide, available at http://www.math.uwaterloo.ca/~clemieux/randqmc.html
36. Lemieux, C., L'Ecuyer, P.: Randomized polynomial lattice rules for multivariate integration and simulation. SIAM Journal on Scientific Computing **24**(5), 1768–1789 (2003)
37. Marion, P., Godin, M., L'Ecuyer, P.: An algorithm to compute the t-value of a digital net and of its projections. Journal of Computational and Applied Mathematics **371**(June), 112669 (2020)
38. Matoušek, J.: On the L_2-discrepancy for anchored boxes. Journal of Complexity **14**, 527–556 (1998)

39. Niederreiter, H.: Low-discrepancy point sets obtained by digital constructions over finite fields. Czechoslovak Math. Journal **42**, 143–166 (1992)
40. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods, *SIAM CBMS-NSF Reg. Conf. Series in Applied Mathematics*, vol. 63. SIAM (1992)
41. Nuyens, D.: Fast component-by-component constructions. http://people.cs.kuleuven.be/~dirk.nuyens/fast-cbc/ (2012). URL http://people.cs.kuleuven.be/~dirk.nuyens/fast-cbc/
42. Nuyens, D.: The construction of good lattice rules and polynomial lattice rules. In: P. Kritzer, H. Niederreiter, F. Pillichshammer, A. Winterhof (eds.) Uniform Distribution and Quasi-Monte Carlo Methods: Discrepancy, Integration and Applications, pp. 223–255. De Gruyter (2014)
43. Nuyens, D.: The magic point shop (2020). https://people.cs.kuleuven.be/~dirk.nuyens/qmc-generators/
44. Nuyens, D., Cools, R.: Fast algorithms for component-by-component construction of rank-1 lattice rules in shift-invariant reproducing kernel Hilbert spaces. Mathematics of Computation **75**, 903–920 (2006)
45. Owen, A.B.: Monte Carlo variance of scrambled equidistribution quadrature. SIAM Journal on Numerical Analysis **34**(5), 1884–1910 (1997)
46. Owen, A.B.: Scrambled net variance for integrals of smooth functions. Annals of Statistics **25**(4), 1541–1562 (1997)
47. Owen, A.B.: Latin supercube sampling for very high-dimensional simulations. ACM Transactions on Modeling and Computer Simulation **8**(1), 71–102 (1998)
48. Owen, A.B.: Variance with alternative scramblings of digital nets. ACM Transactions on Modeling and Computer Simulation **13**(4), 363–378 (2003)
49. Sinescu, V., L'Ecuyer, P.: Variance bounds and existence results for randomly shifted lattice rules. Journal of Computations and Applied Mathematics **236**, 3296–3307 (2012)
50. Sloan, I.H., Joe, S.: Lattice Methods for Multiple Integration. Clarendon Press, Oxford (1994)
51. Sloan, I.H., Rezstov, A.: Component-by-component construction of good lattice rules. Mathematics of Computation **71**, 262–273 (2002)
52. 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**(4), 86–112 (1967)
53. Sobol, I.M., Asotsky, D.I.: One more experiment on estimating high-dimensional integrals by quasi-Monte Carlo methods. Mathematics and Computers in Simulation **62**(3-6), 255–263 (2003)
54. Yue, R.X., Hickernell, F.J.: The discrepancy and gain coefficients of scrambled digital nets. Journal of Complexity **18**(1), 135–151 (2002)