Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.

PRE-SCRAMBLED DIGITAL NETS FOR RANDOMIZED QUASI-MONTE CARLO

Pierre L'Ecuyer[1], Youssef Cherkanihassani[1], and Mohamed El Amine Derkaoui[1]

[1]DIRO, Université de Montréal, Montréal, QC, CANADA

ABSTRACT

When using digital nets for randomized quasi-Monte Carlo, the generating matrices are usually randomly scrambled before applying a random digital shift. This scrambling is to remove the excessive structure that the original matrices may have. In this paper, we explore the idea of pre-scrambling the generating matrices to "optimize" them in some way so that applying the random digital shift alone becomes sufficient. For the optimization criterion, we experiment with a class of recently-proposed figures of merit based on truncated versions of error and variance bounds obtained by bounding the coefficients in the Walsh expansion of smooth integrands. We summarize our numerical experiments and some difficulties encountered.

1 INTRODUCTION

The Monte Carlo (MC) method is used routinely to estimate the mathematical expectation of a function f of several random variables, written as a multivariate integral. By a change of variables, this integral can be expressed as an integral over $(0,1)^s$, the unit hypercube in s dimensions, which can be interpreted as a mathematical expectation with respect to the independent uniform random numbers over the interval $(0,1)$ that drive the simulation. Standard MC draws n independent random points uniformly over $(0,1)^s$, evaluates f at each point, and takes the average to estimate the integral. This average is an unbiased estimator whose variance converges as $\mathcal{O}(n^{-1})$ when $n \to \infty$, so the width of a confidence interval on the mean (which is proportional to the standard deviation) converges as $\mathcal{O}(n^{-1/2})$, which is slow.

Variance reduction methods can reduce the hidden constant in the $\mathcal{O}(n^{-1/2})$ expression, most often without changing the rate (Asmussen and Glynn 2007; L'Ecuyer 2023). In this paper, we are interested in *Quasi-Monte Carlo* (QMC) and *Randomized quasi-Monte Carlo* (RQMC) methods, which under certain regularity conditions do improve the convergence rate (Niederreiter 1992; Dick and Pillichshammer 2010; L'Ecuyer 2009). With QMC, the n independent random points used by MC are replaced by n deterministic points that cover the unit hypercube $(0,1)^s$ more evenly that typical independent random points. The average of the n function evaluations gives a deterministic approximation of the mean. With RQMC, the QMC points are randomized in a way that: (i) each randomized point has the uniform distribution over the unit hypercube whereas (ii) the point set as a whole remains very evenly distributed over the unit hypercube (L'Ecuyer and Lemieux 2002; Owen 2003; L'Ecuyer 2018). The RQMC estimator is again the average of the n evaluations of f. The first condition ensures that this RQMC estimator is unbiased, while the second can make its variance converge at a faster rate than with MC, under appropriate conditions that depend on the way the points are constructed. Here, the n function evaluations are not independent, so the variance cannot be estimated by their sample variance. It is usually estimated by the sample variance of r independent replicates of the RQMC estimator, for a small r.

The two main construction methods for QMC points are lattice rules and digital nets. To define a *lattice rule* of rank 1 with n points in s dimensions, one selects a vector of integers $\boldsymbol{a} = (a_1, \ldots, a_s)$ and the lattice point set is simply $P_n = \{\boldsymbol{u}_i = (i/n)\boldsymbol{a} \bmod 1,\ i = 0, \ldots, n-1\}$. The choice of $\boldsymbol{a}$ determines the quality of the point set (Sloan and Joe 1994; L'Ecuyer and Lemieux 2000; L'Ecuyer and Munger 2012; L'Ecuyer and Munger 2016; Dick et al. 2023). To define a *digital net* in base b (usually $b = 2$, or a prime number), in s dimensions, one selects s *generating matrices* $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_s$ with $w \geq k$ rows and k columns, whose elements are integers in $\{0,\dots,b-1\}$, and whose first k rows are linearly independent in arithmetic modulo b. The QMC point set has $n=b^k$ points and is defined as $P_n=\{\boldsymbol{u}_i=(u_{i,1},\dots,u_{i,s}),\ i=0,\dots,n-1\}$ where $u_{i,j}=\sum_{\ell=1}^{w}u_{i,j,\ell}b^{-\ell}$, and

$$\begin{pmatrix} u_{i,j,1} \\ \vdots \\ u_{i,j,w} \end{pmatrix} = \boldsymbol{C}_j \begin{pmatrix} a_{i,0} \\ \vdots \\ a_{i,k-1} \end{pmatrix} \bmod b, \qquad \text{with } i=\sum_{\ell=0}^{k-1} a_{i,\ell}b^{\ell}.$$

The choice of generating matrices determines the quality of the net. See Niederreiter (1992), Dick and Pillichshammer (2010), L'Ecuyer et al. (2022) for further details. The most popular construction is that of Sobol' (1967) (in base $b=2$) with the direction numbers given by Joe and Kuo (2008). It is popular because efficient implementations are available in many software libraries and environments, and work for arbitrary k and s. For these constructions, each $\boldsymbol{C}_j$ has an infinite number of columns (defined by a recurrence) and is truncated to k columns to get 2^k points, for any $k>0$. Increasing k by 1 doubles the number of points by adding $k/2$ new points to the previous ones. We can also truncate s to the required number of dimensions. When using a digital net in base 2, we usually store the generating matrices by using one w-bit integer for each column. This requires sk integers in total. There are very fast algorithms to compute any point coordinate directly from that. Storing all the point coordinates explicitly would require $s2^k$ integers or floating-point numbers, but we prefer to avoid that because it is too inefficient.

The standard way of randomizing lattice points without destroying their structure, to obtain RQMC points, is a *random shift modulo 1*: generate a single random point in the unit hypercube and add it to all the lattice points, modulo 1. This randomization does not suit digital nets because it does not preserve the equidistribution and t-value properties of the point sets (see Section 2.1). An simple approach that preserves these properties is to replace the random shift modulo 1 by a *random digital shift* (RDS) (L'Ecuyer and Lemieux 1999; L'Ecuyer and Lemieux 2002; Dick and Pillichshammer 2010; L'Ecuyer 2018): generate again a single random point in the unit hypercube and add the digits of the base-b expansion of its coordinates to the corresponding digits of each point, modulo b. For $b=2$ (the most common base), this amounts to making a XOR (exclusive-or) of the binary digits of the random point with the corresponding binary digits of all the points of the digital net. This is computationally very fast and should suffice to obtain good RQMC points if the generating matrices of the net have been carefully constructed.

However, for the commonly available constructions, the generating matrices often have too much structure. For the Sobol' points with the original generating matrices, for example, the projections of the points on certain subsets of high-order coordinates have very poor uniformity (Schmid 2001; Joe and Kuo 2008; Wiart et al. 2021). For this reason, it is common practice to randomize the digital nets by applying more extensive random transformations than just a digital shift. The prominent methods either scramble directly the digits of the point coordinates by random permutations, as in the *nested uniform scramble* (NUS) of Owen (1995), or "scramble" the generating matrices $\boldsymbol{C}_j$ before applying the RDS, as with the *left matrix scramble* (LMS) of Matousěk (1998). LMS left-multiplies (modulo 2) each matrix $\boldsymbol{C}_j$ by a random $w\times k$ matrix $\boldsymbol{L}_j$ whose first k rows form a non-singular lower-triangular submatrix with ones on the diagonal, the entries below the diagonal are independent random bits, and $w\geq k$. See Section 3 for an illustration. With LMS alone, each point *does not* have the uniform distribution (e.g., the point $\mathbf{0}$ is unchanged), but by adding a RDS after the LMS (LMS+RDS) we obtain a valid RQMC scheme. Other linear scrambles of this type are discussed by Matoušek (1999) and Owen (2003). They preserve the digital net property: they only change the generating matrices, so we still have a digital net after the scramble. After applying NUS, on the other hand, the point set is no longer defined as a digital net in general, so it is no longer sufficient to store the generating matrices. The points are usually stored explicitly and this takes more space. The Sobol' points are most often randomized by LMS+RDS, presumably because this is fast and easily accessible in popular software such as MATLAB, R, Python, C++, Java, etc.

Applying NUS or LMS is also motivated and justified by the fact that with these methods, when the integrand f is sufficiently smooth, the variance of the RQMC estimator is guaranteed to converge as $\mathscr{O}(n^{-3}(\log n)^{s-1})$ (Owen 1997; Hickernell and Yue 2001; Yue and Hickernell 2002). This rate is not achieved in general when using Sobol' points with only a RDS.

The convergence rates just mentioned for NUS, LMS, and other similar scrambles are proved by an averaging argument: The variance for a given smooth integrand f is bounded by a constant that measures the variation of f, multiplied by the square discrepancy of the distribution of the points with respect to the uniform distribution. The convergence order of the variance bound is proved by showing that the average square discrepancy of the points over all possible realizations of the randomization converges at the given rate. This means that on average, LMS gives very good generating matrices. But if we take the best of many LMS realizations, we should do better than just picking one at random or taking the average. The retained scramble would give "pre-scrambled" generating matrices that we can save and re-use by applying only the RDS. This will be slightly faster and it may provide an RQMC estimator with smaller variance than redoing LMS for each sample. The aim of the present paper is to explore this idea. We focus on digital nets in base $b = 2$ randomized by LMS. To "optimize" (or select) the LMS, we need an appropriate *figure of merit* (FOM) to measure the quality (discrepancy) of the points of the resulting digital net.

When selecting the parameters for Sobol' points, Joe and Kuo (2008) considered only the t-value of the two-dimensional projections. Those projections are known to have good uniformity, but higher-order projections may be poorly behaved. One can use FOMs based on the t-values for richer sets of projections; see Marion et al. (2020), L'Ecuyer et al. (2022), Section 4.1 of L'Ecuyer (2009), and the references given there, but this is computationally more expensive. Moreover, the t-value in general only looks at the worst-case over a very rich family of partitions into rectangular boxes, and it is often impossible to get a very good value for all the partitions. More sensitive measures should look more closely at what happens for all the partitions, not only the worst case. A recently-proposed measure that goes beyond the t-value and certainly deserves further investigation is the C_b measure of quality from Wiart et al. (2021).

Another prominent and popular FOM, used for example in Latnet Builder, is a weighted $\tilde{\mathscr{P}}_\alpha$ with $\alpha = 2$; see L'Ecuyer et al. (2022), pages 7 to 9. We know how to construct generating matrices for which the square of this discrepancy converges as $\mathscr{O}(n^{-2}(\log n)^{s-1})$. The RQMC variance then converges at least at this rate, for sufficiently smooth functions. But this measure turns out to be *LMS-scramble-invariant*, which means that applying any LMS to the generating matrices of a given digital net does not change their $\tilde{\mathscr{P}}_2$ discrepancy. Therefore, this FOM is not useful for what we want to do. Moreover, for a digital net with $n = 2^k$ points, this FOM depends only on the first k bits of each coordinate of the points. In Section 2.3, we illustrate numerically that LMS is more effective when we randomize more than the first k bits. This suggests that a good FOM should look at more than just the first k bits.

In this paper, we consider alternative FOMs known as *Walsh figures of merit* (WAFOMs), which are based on the fact that the QMC integration error with a digital net can be written as the sum of Walsh coefficients of f over the dual net, and that bounds on these coefficients are available for smooth functions (Dick 2009; Dick and Pillichshammer 2010; Yoshiki 2017). For RQMC with a RDS only, the variance is equal to the sum of squared Walsh coefficients over the same space (L'Ecuyer and Lemieux 2002; Lemieux and L'Ecuyer 2003), and similar WAFOMs are also available. In principle, the sum of the bounds on the coefficients (squared for RQMC) for a given class of integrands could be taken as a FOM. But these sums have an infinite number of terms. Matsumoto et al. (2014) proposed to approximate f by a piecewise-constant function with a very large number of small cubic pieces to truncate the sum to a finite number of terms, and also derived an alternative expression for this finite sum with a much smaller number of terms, for the deterministic case. They called this expression the "WAFOM." Yoshiki (2017) proposed a slightly different version, based on different bounds. Harase (2015, 2016) discusses their efficient implementation and the search for low-WAFOM point sets. Goda et al. (2016) proposed modified versions based on approximate bounds for the RQMC variance. When $w > k$, these WAFOMs depend on all the w bits of all coordinates of all points. Our Section 3 provides a review of all these WAFOMs.

One way to search for low-WAFOM digital nets is to start with Sobol' generating matrices, apply several independent LMS scrambles to them, and retain the ones with the lowest WAFOMs. If s and k

are fixed, each LMS can be fully generated for all coordinates before selecting the best one. But it is practically more convenient when the same point sets can be used (by truncating or extending) for any k and s. Constructing extensible digital nets using WAFOMs turns out to be more challenging. Section 4 discusses different ways of implementing this optimization process (one coordinate at a time, one matrix column at a time, etc.). We report experimental results in Section 5. A brief conclusion follows.

2 RQMC WITH DIGITAL NETS, WITH LMS AND RDS

Here we recall some definitions and properties of digital nets. More details can be found in Dick and Pillichshammer (2010), L'Ecuyer (2018), and the references given there. We also give numerical illustrations that compare different ways of randomizing a digital net, in terms of the resulting RQMC variance.

2.1 Equidistribution and t-Value

A key property of a digital net P_n is that the base-b digits of the coordinates of $\boldsymbol{u}_i$ are obtained by a linear transformation of the base-b digits of the integer i. In particular, the first q_j digits of the jth coordinate $u_{i,j}$ are determined by the linear transformation defined by the first q_j rows of $\boldsymbol{C}_j$. If we divide each axis j in b^{q_j} equal parts, for some integers $q_j \geq 0$, we obtain a partition of $[0,1)^s$ into $b^q = b^{q_1+\cdots+q_s}$ rectangular boxes of equal sizes, and the box in which point $\boldsymbol{u}_i$ falls is determined by the linear transformation defined by the matrix whose rows are the first q_j rows of $\boldsymbol{C}_j$, for $j = 1,\ldots,s$, in arithmetic modulo b. Basic linear algebra tells us that each box will contain the same number of points from P_n if and only if this matrix has rank q, which is of course possible only if $q \leq k$. We the say that the points are $(q_1,\ldots,q_s)$*-equidistributed in base b*. This property is easy to verify, just by computing the rank of a matrix. As a special case, by taking $q_j = q = k$, we find that coordinate j truncated to its first k digits will visit all b^k possible values exactly once if and only if the first k rows of $\boldsymbol{C}_j$ are linearly independent (modulo b). This is a basic property that we want to keep when constructing or randomizing the matrices $\boldsymbol{C}_j$.

A point set P_n is called a (t,k,s)*-net in base b* if and only if it is $(q_1,\ldots,q_s)$-equidistributed whenever $q_1+\cdots+q_s = k-t$. This is possible for $t = 0$ only if $b \geq s-1$. The t*-value* of a digital net is the smallest t for which it is a (t,k,s)-net. An important result that connects the t-value with QMC error bounds is that for fixed s and t, if P_n is a (t,k,s)-net in base b for $k = 1,2,3,\ldots$, then the star discrepancy of P_n converges as $\mathscr{O}(n^{-1}(\log n)^{s-1})$, and for any f having finite Hardy-Krause variation, the QMC integration error with these point sets converges at this same rate. Moreover, for randomizations that preserve the (t,k,s)-net property, the RQMC variance converges as $\mathscr{O}(n^{-2}(\log n)^{2(s-1)})$.

LMS and NUS preserve the equidistribution and t-value: If a digital net is $(q_1,\ldots,q_s)$-equidistributed, then it remains $(q_1,\ldots,q_s)$-equidistributed after applying LMS or NUS. LMS also preserves the digital net property and is usually faster to apply than NUS, hence our preference.

2.2 Experimental Setting and Test Functions

The numerical experiments reported in this paper are for the following four test functions, all defined over the unit hypercube in s dimensions. The first three are from Genz (1987). The fourth one is taken from Section 7.3 of L'Ecuyer et al. (2022); it generalizes a function used by Sobol and Asotsky (2003) who had $a_j = c$ for all j. In the following, $\boldsymbol{u} = (u_1,\ldots,u_s)$ denotes a point in $[0,1)^s$, and φ and Φ are the density and the cdf of the standard normal distribution. We take $a_j = j/s$ and $s = 3$, 6, 12 for all functions.

1. `Oscillatory`: $f(\boldsymbol{u}) = \cos\left(\sum_{j=1}^s a_j u_j\right)$.
2. `Exponential`: $f(\boldsymbol{u}) = \exp\left((2/3)\sum_{j=1}^s u_j\right)$.
3. `Gaussian`: $f(\boldsymbol{u}) = \exp\left(\sum_{j=1}^s u_j^2\right)$.
4. `Polynomial`: $f(\boldsymbol{u}) = \prod_{j=1}^s (1 + a_j \cdot (u_j - 1/2))$.

We also used other functions from Genz (1987) and L'Ecuyer et al. (2023) in our experiments and our summaries account for all these additional experiments.

2.3 Does LMS Really Reduces the Variance?

In the rest of the paper, we assume that $b = 2$. For all our experiments, the initial point sets were Sobol' nets with the direction numbers of Joe and Kuo (2008). The variance bound of $\mathcal{O}(n^{-3}(\log n)^{s-1})$ with NUS or LMS was proved under the assumption that these methods are applied with $w = \infty$. If w is too small, or if we do only the RDS, we may get a slower rate. To see this and get a sense of the impact of performing the additional LMS rather than just apply a RDS alone, we looked at how the variance behaves with: (i) only a RDS (RDS); (ii) LMS with $w = k$ followed by RDS (LMS-k); (iii) LMS with $w = 31$ followed by RDS (LMS-31). The RDS is always done with a 31-bit digital shift. For each test function of Section 2.2 in $s = 3$, 6, 12 dimensions, We made $r = 1000$ independent replications of the RQMC process to estimate the variance for $k = 8, \ldots, 20$, and plotted the log of this variance as a function of k. For each method, the observations at the different values of k are independent. We also did ordinary MC for comparison. We found that (i) and (ii) give almost the same RQMC variance, (iii) is usually better and sometimes much better, depending on f and its dimension, and all three give a better rate than MC. Figure 1 shows three cases, with $s = 6$. The orange and green lines indicate rates of $\mathcal{O}(n^{-2})$ and $\mathcal{O}(n^{-3})$, for comparison. The slope of the red path approaches that of the green line for the first two functions. For the Gaussian, the LMS does not seem to help; this happens in some cases. The orange path (wfixed) is for a particular LMS constructed by the "FIXED" method described in Section 5. It outperforms the randomized LMSs.

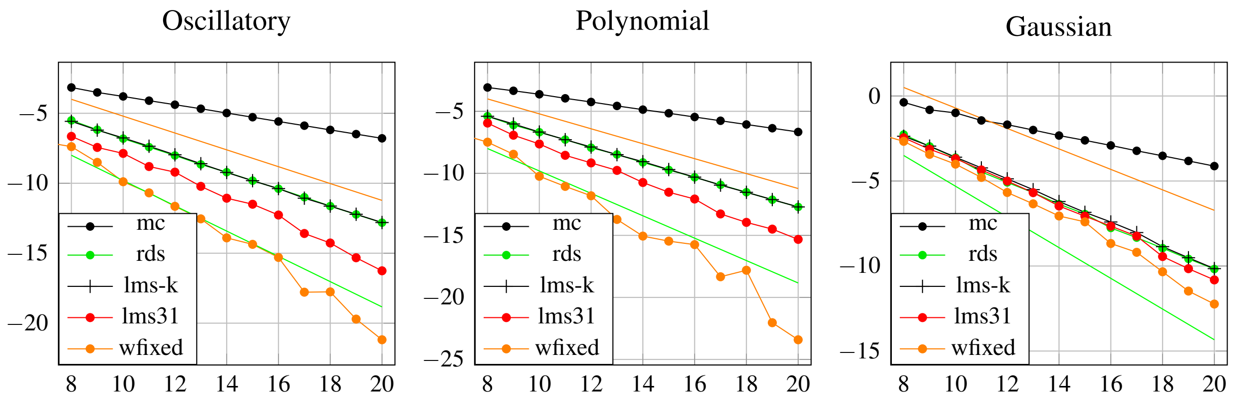

Figure 1: Log_{10} of the Variance as a function of k with five methods, for three functions, with $s = 6$.

2.4 Comparing Scrambling Realizations

Suppose we apply LMS+RDS to a digital net to estimate the integral of f with RQMC. Let $\bar{X}_{\text{rqmc}}$ be the RQMC estimator, v_f its unconditional variance, L denote the realization of the LMS (i.e., of $\boldsymbol{L}_1, \ldots, \boldsymbol{L}_s$), and $v_f(L) = \text{Var}[\bar{X}_{\text{rqmc}} \mid L]$. We have $v_f = \mathbb{E}[v_f(L)]$, because $\text{Var}[\mathbb{E}[\bar{X}_{\text{rqmc}} \mid L]] = 0$. This means that unless all $v_f(L)$ are the same, there is at least one L for which $v_f(L) < v_f$. Picking such a L and applying only RDS would give a lower-variance RQMC estimator than doing the random LMS.

We made experiments to get an idea of the distribution of $v_f(L)$ for our test functions f, for $s = 3$, 6, and 12, and to see if the good L's for one f tends to be also good for the other functions. If this holds, then it would make sense to search for good L's, at least for certain classes of functions. For each considered (f, s, k), we draw 1000 realizations of L and made 200 independent RDSs for each one to estimate $v_f(L)$.

The distribution of $v_f(L)$ varies depending on f, s, and also k. Figure 2 shows three cases, for $s = 6$ and $k = 16$. For the Polynomial function, the observations of $v_f(L)$ range approximately from 10^{-25} to 10^{-10},

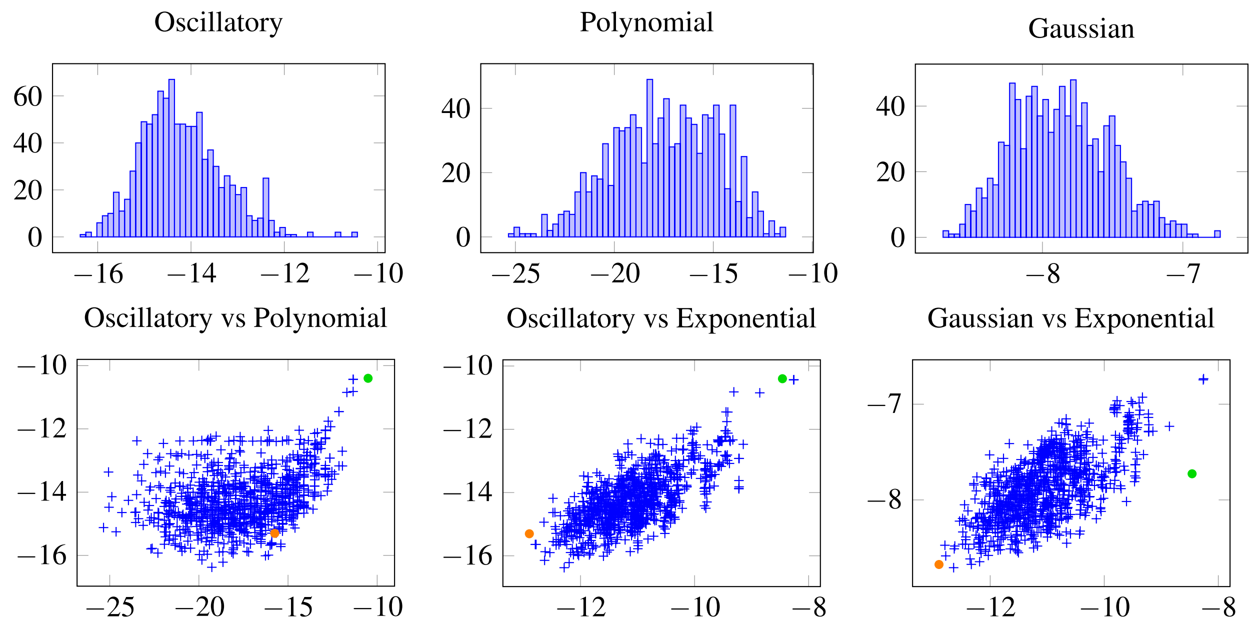

Figure 2: Above: Histograms of 1000 realizations of $\log_{10}(v_f(L))$ for three functions, for $s = 6$ and $k = 16$. Below: Scatter plots of the 1000 points $(\log_{10}(v_{f_1}(L)), \log_{10}(v_{f_2}(L)))$, for three pairs of functions. The green and orange dots represent $\log_{10}(v_f)$ for RDS-only and for the "FIXED" method of Section 5.

so the ratio between the worst and best $v_f(L)$ is about 10^{15}, which is huge! There are also cases where this factor is about 10. The scatter plots suggest a positive dependence between the variances for pairs of functions. We also found cases where the dependence is weaker. The green dots show that applying LMS is very helpful for the Oscillatory, Polynomial, and Exponential functions, but not much for the Gaussian function, as we already saw in Figure 1. Note that $\mathbb{E}[\log_{10}(v_f(L)] < \log_{10}\mathbb{E}[v_f(L)]$. For the Polynomial function with LMS+RDS, for example, the average $\log_{10}(v_f(L))$ is -17.457 while $\log_{10}(v_f)$ is -12.07.

3 WALSH FIGURES OF MERIT

When using a digital net in base 2 for QMC integration of a function f, the (deterministic) error is given by $\sum_{\mathbf{0}\neq\boldsymbol{k}\in\mathscr{D}} \hat{f}(\boldsymbol{k})$, where $\mathscr{D} \subset \mathbb{N}_0^s$ is the dual net and the $\hat{f}(\boldsymbol{k})$ for $\boldsymbol{k} = (k_1, \ldots, k_s) \in \mathbb{N}_0^s$ are the coefficients in the Walsh expansion of f, under the assumption that $\sum_{\mathbf{0}\neq\boldsymbol{k}\in\mathscr{D}} |\hat{f}(\boldsymbol{k})| < \infty$. Moreover, the RQMC variance with a RDS is equal to $\sum_{\mathbf{0}\neq\boldsymbol{k}\in\mathscr{D}} |\hat{f}(\boldsymbol{k})|^2$ whenever this sum is finite, i.e., when the RQMC variance is finite. See L'Ecuyer and Lemieux (1999), Lemieux and L'Ecuyer (2003), Dick and Pillichshammer (2010) for more details. Dick (2007, 2008) has shown that if f has smoothness α, which means (very roughly) that all its mixed partial derivatives of order up to α with respect to each coordinate are integrable, then

$$\left|\hat{f}(\boldsymbol{k})\right| \leq K\|f\|_\alpha 2^{-\mu_\alpha(\boldsymbol{k})} \tag{1}$$

approximately, where K is a constant, $\|f\|_\alpha$ is a measure of variation of f that depends on the integrals of the mixed partial derivatives, $\mu_\alpha(\boldsymbol{k}) = \mu_\alpha(k_1) + \cdots + \mu_\alpha(k_s)$, $\mu_\alpha(k) = (a_1+1) + \cdots + (a_{\min(\alpha,v)}+1)$ if $k = 2^{a_1} + \cdots + 2^{a_v} > 0$ with $a_1 > \cdots > a_v$, and $\mu_\alpha(0) = 0$. This suggests using the FOM $\sum_{\mathbf{0}\neq\boldsymbol{k}\in\mathscr{D}} 2^{-\mu_\alpha(\boldsymbol{k})}$ for QMC and $\sum_{\mathbf{0}\neq\boldsymbol{k}\in\mathscr{D}} 2^{-2\mu_\alpha(\boldsymbol{k})}$ for RQMC, for a selected α. But this is impractical because these sums have an infinite number of terms. For this reason, Dick (2009) replaces the first sum by its largest term, $\max_{\mathbf{0}\neq\boldsymbol{k}\in\mathscr{D}} 2^{-\mu_\alpha(\boldsymbol{k})}$, which he wants to minimize. This is a much cruder FOM, but is easier to compute.

Replacing the second sum by its largest term gives an equivalent FOM (the value is squared). Dick then shows how to construct digital nets for which $\max_{\mathbf{0}\neq\boldsymbol{k}\in\mathscr{D}} 2^{-\mu_\alpha(\boldsymbol{k})} = \mathscr{O}(n^{-\alpha}(\log n)^{\alpha s})$, using an interlacing technique. This is implemented in LatNet Builder (L'Ecuyer et al. 2022).

Matsumoto et al. (2014) proposed an alternative FOM that truncates the first sum to the vectors $\boldsymbol{k}$ whose coordinates are all less that 2^w in absolute value, and takes $\alpha = w$, where w represents the number of output digits that are considered. The number of terms in the sum is now finite, but can still be large. With $n = 2^k$, this sum is over 2^{sw-k} binary vectors $\boldsymbol{k}$. For $w = 31$, $s = 20$, and $k = 16$, for example, we have $n = 2^{16}$ points and the sum has 2^{604} terms. But Matsumoto et al. (2014) have shown that the truncated sum can be rewritten equivalently as

$$\mathscr{W}_{\mathrm{M}}(P_n) = -1 + \frac{1}{n}\sum_{i=0}^{n-1}\prod_{j=1}^{s}\prod_{\ell=1}^{w}\left(1+\eta(u_{i,j,\ell})2^{-\ell}\right), \tag{2}$$

where $\eta(u) = (-1)^u$, and $u_{i,j,\ell}$ is digit ℓ of coordinate j of point $\boldsymbol{u}_i \in P_n$. This sum has only $n = 2^k$ terms and is much faster to compute. Matsumoto and Yoshiki (2013) have shown that for $k \geq 9s$, there exist point sets P_n for which $\mathscr{W}_{\mathrm{M}}(P_n) = \mathscr{O}(n^{-C(\log_2 n)/s+D})$ for some constants C and D. But with the condition that $k \geq 9s$, this result is practically relevant only for very small s. It is then useful to explore empirically what rate we can achieve with practical constructions.

Yoshiki (2017) obtained a different bound than Dick by showing that $\mu_\alpha(\boldsymbol{k})$ can be replaced by $\mu'_\alpha(\boldsymbol{k}) = \mu'_\alpha(k_1) + \cdots + \mu'_\alpha(k_s)$ in which $\mu'_\alpha(k) = (a_1+2) + \cdots + (a_{\min(\alpha,\nu)}+2) = \mu_\alpha(\boldsymbol{k}) + \min(\alpha,\nu)$, under the condition that all mixed partial derivatives of order up to α with respect to each coordinate are *continuous*. The constant K in (1) is also replaced by a different constant K' that can be larger than K, so the new bound is not always smaller. This new bound leads to the modified WAFOM:

$$\mathscr{W}_{\mathrm{MY}}(P_n) = -1 + \frac{1}{n}\sum_{i=0}^{n-1}\prod_{j=1}^{s}\prod_{\ell=1}^{w}\left(1+\eta(u_{i,j,\ell})2^{-(\ell+1)}\right). \tag{3}$$

Goda et al. (2016) introduced versions of (2) and (3) that represent bounds on the RQMC variance $\sum_{\mathbf{0}\neq\boldsymbol{k}\in\mathscr{D}} |\hat{f}(\boldsymbol{k})|^2$ when using RDS:

$$\mathscr{W}^2_{\mathrm{G}}(P_n) = -1 + \frac{1}{n}\sum_{i=0}^{n-1}\prod_{j=1}^{s}\prod_{\ell=1}^{w}\left(1+\eta(u_{i,j,\ell})2^{-2\ell}\right), \tag{4}$$

$$\mathscr{W}^2_{\mathrm{GY}}(P_n) = -1 + \frac{1}{n}\sum_{i=0}^{n-1}\prod_{j=1}^{s}\prod_{\ell=1}^{w}\left(1+\eta(u_{i,j,\ell})2^{-2(\ell+1)}\right). \tag{5}$$

These FOMs are more appropriate than the previous ones when we use a RDS for RQMC, because the variance is bounded approximately by a constant times these bounds.

All these WAFOMs can be generalized to cover weighted function spaces with general projection-dependent weights, for which the WAFOM is computed for each subset of coordinates, and the full WAFOM is the weighted sum of those, with a weight for each projection. Yoshiki (2017) derives his formulas in this setting. Most FOMs in LatNet Builder are also implemented for this setting.

In the remainder, the generic term "WAFOM" refers to a general class of measures that includes (2) to (5). What we say about the computation of (2) applies equally well to the other variants.

4 SEARCHING FOR A GOOD LMS IN TERMS OF THE WAFOM

We now explain how the Wafom can be computed efficiently and how it could be used to select good LMSs. When computing (2), each coordinate $u_{i,j}$ is represented as a w-bit integer $z_{i,j} = \lfloor 2^w u_{i,j} \rfloor$, not a floating-point number, and we extract the bits $u_{i,j,\ell}$ from this integer using shifts and masks. The expression

$(1+\eta(u_{i,j,\ell})2^{-\ell})$ has only two possible values for each ℓ, and only $2w$ possible values in total since there are w possible values of ℓ. These $2w$ values can be precomputed and stored in a table to speed up the computations. More speedup can be achieved by precomputing partial products as follows (Harase 2016). Select integers $v_1, v_2, v_3, \ldots$, each around 8 or 10, whose sum is w. For $w = 31$, one can take $v_1 = v_2 = v_3 = 8$ and $v_4 = 7$, for example. The partial product $p_{i,j,1,v_1} = \prod_{\ell=1}^{v_1} \left(1+\eta(u_{i,j,\ell})2^{-\ell}\right)$ can take only 2^{v_1} possible values, depending on $(u_{i,j,1},\ldots,u_{i,j,v_1})$. We can precompute these 2^{v_1} values and store them in a table. Likewise, the partial product $p_{i,j,v_1+1,v_1+v_2} = \prod_{\ell=v_1+1}^{v_1+v_2} \left(1+\eta(u_{i,j,\ell})2^{-\ell}\right)$ can only take 2^{v_2} possible values, which we can precompute and store, and so on. For a given $z_{i,j}$, one would look in the first table for the value of $p_{i,j,1,v_1}$ depending on the first v_1 bits of $z_{i,j}$, then the value of p_{i,j,v_1+1,v_1+v_2} that depends on the next v_2 bits, etc., and multiply those partial products to obtain the full products in (2).

When computing a WAFOM for fixed s, we can just compute the double product in (2) one point at a time, and add up. If we want to construct point sets that are extensible in the dimension, i.e., for which the first s coordinates gives a good s-dimensional point set for any $s \geq 1$, one possible approach could be to use a standard coordinate by coordinate (CBC) construction as follows. When we compute (2) for a given $s-1$, we memorize the n products in the sum. When adding coordinate s, it suffices to compute the inner product for $j = s$ and multiply it by the previously-stored double product, for each point i.

To construct point sets that are extensible in the number of points, i.e., for which we can always increase k by 1 to double the number of points, we would like to be able to add columns one by one to the generating matrices. When adding column k, the previous 2^{k-1} points remain the same and there are 2^{k-1} new points, for a total of 2^k points. To update the WAFOM, it suffices to compute and add the double products in (2) for the new points, and add their sum to the previous sum, because the double products for the old points do not change. When adding a new column k to a generating matrix $\boldsymbol{C}_j$, we want to make sure that the new upper left $k \times k$ submatrix is invertible, as mentioned earlier. For this, if we assume that the property was true for the $(k-1)\times(k-1)$ submatrix, it suffices to check that column k is independent from the previous ones. If it is not, we simply flip the last bit of the new column (at position (k,k)) and it will become independent. Of course, testing the linear independence for each k requires work.

One special case where this testing is not required is when we apply LMS to Sobol' upper-triangular matrices $\boldsymbol{C}_j$ using lower-triangular invertible matrices $\boldsymbol{L}_j$, to obtain the new generating matrices $\tilde{\boldsymbol{C}}_j = \boldsymbol{L}_j\boldsymbol{C}_j$. These $\tilde{\boldsymbol{C}}_j$ inherit the equidistribution properties of the $\boldsymbol{C}_j$. The $\boldsymbol{L}_j$ and $\boldsymbol{C}_j$ have the form:

$$\boldsymbol{L}_j = \boldsymbol{L} = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ \ell_{2,1} & 1 & 0 & \ldots & 0 \\ & \vdots & & \ddots & 0 \\ \ell_{k,1} & \ell_{k,2} & & & 1 \\ \ell_{k+1,1} & \ell_{k+1,2} & & & \ell_{k+1,k} \\ & \vdots & & & \vdots \\ \ell_{w,1} & \ell_{w,2} & & & \ell_{w,k} \end{pmatrix} \quad \text{and} \quad \boldsymbol{C}_j = \boldsymbol{C} = \begin{pmatrix} 1 & v_{1,2} & \ldots & v_{1,k} \\ 0 & 1 & \ldots & v_{2,k} \\ \vdots & 0 & \ddots & \vdots \\ & \vdots & & 1 \end{pmatrix}.$$

One can easily verify that the kth column of $\tilde{\boldsymbol{C}}_j$ depends only on the first k columns of $\boldsymbol{L}_j$, for any k. Therefore, the first k columns of $\tilde{\boldsymbol{C}}_j$ will never change when we add new columns to $\boldsymbol{L}_j$ and $\boldsymbol{C}_j$ (to increase k). When we add column k, we can only select the $w-k$ blue entries (bits) in the display. When trying to select this column to optimize a FOM (with everything else fixed), we may sample several choices at random for the blue entries, and select the best choice. Since the first $k-1$ columns of $\tilde{\boldsymbol{C}}_j$ do not depend on column k of $\boldsymbol{L}_j$, they will remain unchanged, so the first 2^{k-1} points will remain unchanged. In fact, changing the bit $\ell_{r,k}$ from 0 to 1 just flips the corresponding bit $\tilde{c}_{r,k}$ in the matrix $\tilde{\boldsymbol{C}}_j$ and has no other effect. This means that picking the bits $\ell_{k+1,k},\ldots,\ell_{w,k}$ at random is equivalent to directly picking the bits $\tilde{c}_{k+1,k},\ldots,\tilde{c}_{w,k}$ of the matrix $\tilde{C}_j$ at random. On the other hand, the first k bits of column k depend on the previous columns of $\boldsymbol{L}_j$, so we need these previous columns to compute the top k bits.

A natural way to construct low-WAFOM digital nets that are extensible in both s and k could be to start with a given digital sequence (e.g., Sobol' points) and apply LMS incrementally as follows. Select a maximal dimension $s_{\max}$ and two integers $k_{\max} > k_{\min} > 0$. For $j = 1, 2, \ldots, s_{\max}$ sample N matrices $\boldsymbol{L}_j$ with $k = k_{\min}$ columns, compute (update) the WAFOM for the current j-dimensional 2^k-point net for each sample, and selects the $\boldsymbol{L}_j$ that gives the smallest WAFOM. Then, for $k = k_{\min}+1, \ldots, k_{\max}$ and $j = 1, \ldots, s_{\max}$, add column k to $\boldsymbol{L}_j$ by sampling N random choices and retain the one that gives the best updated WAFOM. We call this Algorithm EXT-two (for *two-way extensible*).

We also define the following variants. Algorithm EXT-dim constructs a net that is extensible only in the dimension, for fixed k. At each step, we add a new coordinate j by sampling N possibilities for L_j (all columns at once) and retaining the best. Algorithm EXT-size constructs a net that is extensible only in the number of points, for a fixed dimension s. At each step, we increase k by 1 by sampling N possibilities for the column k of all the matrices L_j at once, and retaining the sample that gives the best WAFOM. Algorithm FIXED constructs a net that is not extensible at all. We fix both s and k, and for each of the N trials, we sample all the columns of all the matrices $\boldsymbol{L}_j$ at once, and compute the WAFOM.

Instead of applying LMS to an existing net, it is also possible to sample the columns of the generating matrices directly at random, with any of the four extensibility options described above. This can be done by making sure that the $k \times k$ upper-left submatrices are always invertible, for all k, or by ignoring that.

Some of these construction methods have been explored by previous authors. We now summarize this previous work. Harase (2015) explored Algorithm FIXED in an example in which he starts with a known digital net with good t-value (he tried both Sobol' and Niederreiter-Xing nets) in $s = 5$ dimensions, with $w = 32$, each k from 1 to 23, using $N = 10^5$ random trials for each k. He also tried generating the entries of the generating matrices directly at random, for comparison. He reports the WAFOM $\mathscr{W}_{\mathrm{MY}}(P_n)$ in (3) and the (deterministic) QMC integration error for selected functions f taken from Genz (1987), as a function of k. The best WAFOMs obtained by the three methods were very similar, better than those of the original Niederreiter-Xing nets, and much better than those of the unscrambled Sobol' nets. The integration error was highly reduced and converged at a faster rate with the low-WAFOM constructions than with the original Sobol' points for smooth integrands f. For discontinuous or non-smooth f, his experiments suggest that the t-value of the net is more important than the WAFOM. He did not try RQMC.

Harase (2016) (written in 2013) constructed low-WAFOM nets by sampling directly the columns of the generating matrices for all dimensions at once, for $s = 5$ (only) and $w = 32$, to construct digital nets that are extensible in the number of points (EXT-size), for $k = 8, \ldots, 25$. He used the WAFOM for QMC in (3), and up to $N = 10^5$ random trials for each k. He examines the same quantities as in Harase (2015) and his results are very similar. Earlier, Matsumoto et al. (2014) searched for low-WAFOM point sets in the class of recurrence-based digital nets constructed by $\mathbb{F}_2$-linear recurrences as in L'Ecuyer and Lemieux (1999). However, Harase (2016) obtained smaller WAFOMs.

Goda et al. (2016) made similar experiments to search for low-WAFOM point sets in terms of $\mathscr{W}_{\mathrm{GY}}(P_n)$ in (5), for RQMC. They tried sampling all the generating matrices at random, with $N = 1000$ trials, for $s = 4$ and $s = 12$ (fixed s and fixed k). They also tried a simulated annealing approach to "optimize" the WAFOM. In experiments similar to Harase (2015) but for RQMC, they observed a strong positive correlation between their WAFOM (5) and the RQMC variance for smooth functions.

5 RQMC EXPERIMENTS WITH WAFOM-BASED DIGITAL NET CONSTRUCTIONS

To test and compare the methods outlined in the previous section, we implemented in SSJ (L'Ecuyer and Buist 2005) efficient algorithms to compute the WAFOMs (2) to (5) and the four algorithms FIXED, EXT-dim, EXT-size, and EXT-two, to search for LMSs that give good WAFOMs. We ran the three "EXT" methods for $N = 10$, 20, 50, 300, and retained the best extensible construction in each case. For the FIXED method, for each (s,k) we sampled 10,000 independent LMS's and retained the best. We found that the "EXT" methods give significantly larger WAFOM values than the FIXED one. Figure 3 illustrates this by showing the values of (3) or (5) obtained by the four LMS construction methods, as functions of k, for

$w = 31$, $k_{\min} = 6$, $k_{\max} = 20$, $s_{\max} = 6$, for three cases. The other cases are similar. We used the given value of N for the "EXT" methods, and $N = 10,000$ for the FIXED method. Increasing N for the "EXT" methods does not bring much improvement. We see that fixing the generating matrices coordinate by coordinate or column by column without going back does not work well with the WAFOM, in contrast with the CBC methods commonly employed to construct extensible QMC point sets with other criteria (Dick and Pillichshammer 2010; L'Ecuyer and Munger 2016; L'Ecuyer et al. 2022). For the WAFOMs, the early choices have a lot of impact on the future WAFOM values (for larger s or k), and the best choices for the current (s,k) are often bad for the future values. In fact, the first k columns of $\boldsymbol{L}$ already determine the first k bits of all the future points, so they largely determine the WAFOMs for all larger values of k. We also tried modified construction algorithms that retain a small population of constructions at each step rather than only the one with lowest WAFOM, but the results were only slightly better.

It then appears preferable to use a different LMS for each pair (s,k). A collection of digital nets selected via the FIXED method, one for each pair (s,k) in a given range, can be stored in files for general use. Another approach that does almost as well in our preliminary experiments and gives extensible nets is similar to the FIXED method is the following. For a given s, we generate N LMSs for the largest k of interest, but instead of computing the WAFOM only for that k, we compute it for a range of smaller values of k, look at the ratio between that value and the best one obtained for that k, and take the worst case with respect to k as a FOM. We plan to study this approach more extensively in future work.

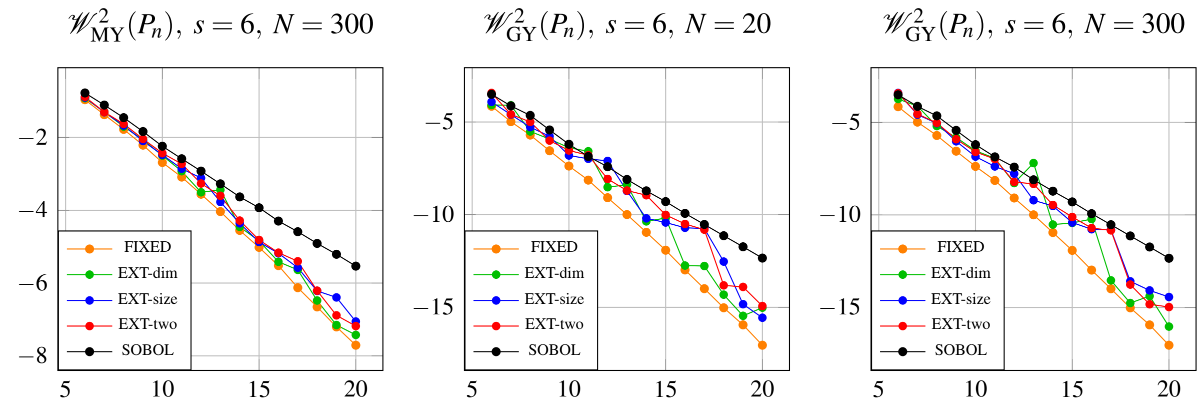

Figure 3: Log of WAFOM as a function of k for $s = 6$ with different construction algorithms for (3) with $N = 300$ (left), then (5) with $N = 20$ and $N = 300$ (center and right). SOBOL is for Sobol' points directly.

Figure 4 shows the log of the RQMC variance with RDS only, for each selected construction and with the original Sobol' points. The FIXED case (in orange) is more consistent than the "EXT" ones. It also gives a much smaller variance and a better rate (empirically) than random LMS + RDS (lms31); see Figure 1. That is, the idea of optimizing the LMS appears quite effective.

6 CONCLUSION

We gave a review of WAFOMs and examined their possible use to improve the generating matrices of digital nets by trying to "optimize" the LMS. The results are very encouraging. More work is needed to improve the search methods for extensible nets. We also plan to implement in LatNet Builder (L'Ecuyer et al. 2022) weighted versions of the WAFOM, with arbitrary weights on subsets of coordinates.

ACKNOWLEDGMENTS

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), Grant RGPIN-2024-05864. Yocheved Darmon made preliminary experiments as in Section 2.4 in 2021.

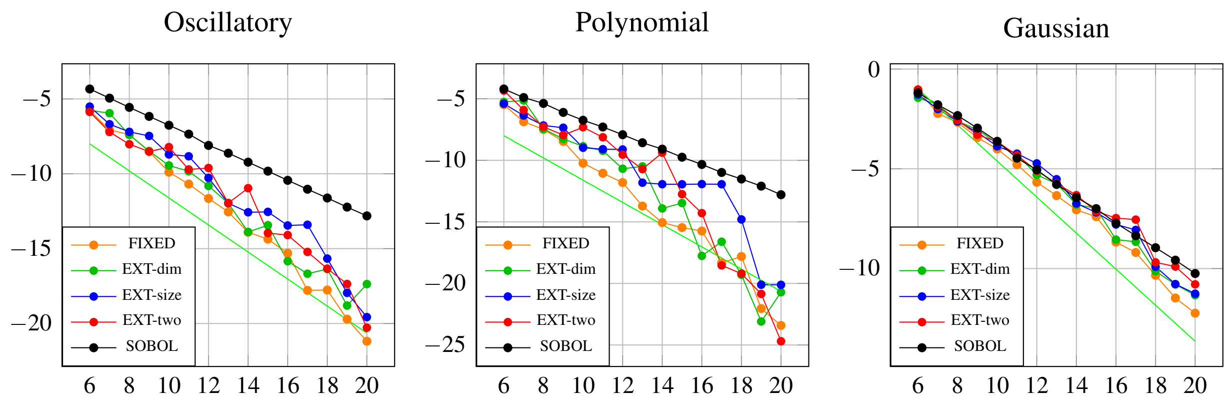

Figure 4: Log of RQMC variance as a function of k for the retained construction based on the WAFOM (5), with RDS only, for $s = 6$ and $N = 20$. The green solid line has slope $\mathscr{O}(n^{-3})$.

REFERENCES

Asmussen, S. and P. W. Glynn. 2007. *Stochastic Simulation*. New York: Springer-Verlag.

Dick, J. 2007. "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.

Dick, J. 2008. "Walsh Spaces Containing Smooth Functions and Quasi-Monte Carlo Rules of Arbitrary High Order". *SIAM Journal on Numerical Analysis* 46(3):1519–1553.

Dick, J. 2009. "On Quasi-Monte Carlo Rules Achieving Higher Order Convergence". In *Monte Carlo and Quasi-Monte Carlo Methods 2008*, edited by P. L'Ecuyer and A. B. Owen, 73–96. Berlin: Springer-Verlag.

Dick, J., P. Kritzer, and F. Pillichshammer. 2023. *Lattice Rules: Numerical Integration, Approximation, and Discrepancy*. Springer.

Dick, J. and F. Pillichshammer. 2010. *Digital Nets and Sequences: Discrepancy Theory and Quasi-Monte Carlo Integration*. Cambridge, U.K.: Cambridge University Press.

Genz, A. 1987. "A Package for Testing Multiple Integration Subroutines". In *Numerical Integration: Recent Developments, Software and Applications*, edited by P. Keast and G. Fairweather, 337–340. Dordrecht: Springer Netherlands.

Goda, T., R. Ohori, K. Suzuki, and T. Yoshiki. 2016. "The Mean Square Quasi-Monte Carlo Error for Digitally Shifted Digital Nets". In *Monte Carlo and Quasi-Monte Carlo Methods 2014*, edited by R. Cools and D. Nuyens, 331–350. Berlin: Springer-Verlag.

Harase, S. 2015. "Quasi-Monte Carlo point sets with small t-values and WAFOM". *Applied Math. and Computation* 254:318–326.

Harase, S. 2016. "A search for extensible low-WAFOM point sets". *Monte Carlo Methods and Applications* 22(4):349–357.

Hickernell, F. J. and R.-X. Yue. 2001. "The Mean Square Discrepancy of Scrambled (t,s)-Sequences". *SIAM Journal on Numerical Analysis* 38(4):1089–1112.

Joe, S. and F. Y. Kuo. 2008. "Constructing Sobol Sequences with Better Two-Dimensional Projections". *SIAM Journal on Scientific Computing* 30(5):2635–2654.

L'Ecuyer, P. 2009. "Quasi-Monte Carlo Methods with Applications in Finance". *Finance and Stochastics* 13(3):307–349.

L'Ecuyer, P. 2018. "Randomized Quasi-Monte Carlo: An Introduction for Practitioners". In *Monte Carlo and Quasi-Monte Carlo Methods: MCQMC 2016*, edited by P. W. Glynn and A. B. Owen, 29–52. Berlin: Springer.

L'Ecuyer, P. 2023. "Stochastic Simulation and Monte Carlo Methods". Draft Textbook, https://www-labs.iro.umontreal.ca/~lecuyer/ift6561/book.pdf, accessed 16th August 2024.

L'Ecuyer, P. and E. Buist. 2005. "Simulation in Java with SSJ". In *2005 Winter Simulation Conference (WSC)*, 611–620 https://dl.acm.org/doi/abs/10.5555/1162708.1162815.

L'Ecuyer, P. and C. Lemieux. 1999. "Quasi-Monte Carlo via Linear Shift-Register Sequences". In *1999 Winter Simulation Conference (WSC)*, 632–639 https://doi.org/10.1145/324138.324448.

L'Ecuyer, P. and C. Lemieux. 2000. "Variance Reduction via Lattice Rules". *Management Science* 46(9):1214–1235.

L'Ecuyer, P. and C. Lemieux. 2002. "Recent Advances in Randomized Quasi-Monte Carlo Methods". In *Modeling Uncertainty: An Examination of Stochastic Theory, Methods, and Applications*, edited by M. Dror, P. L'Ecuyer, and F. Szidarovszky, 419–474. Boston: Kluwer Academic.

L'Ecuyer, P., P. Marion, M. Godin, and F. Puchhammer. 2022. "A Tool for Custom Construction of QMC and RQMC Point Sets". In *Monte Carlo and Quasi-Monte Carlo Methods: MCQMC 2020*, edited by A. Keller, 51–70. Berlin: Springer.

L'Ecuyer, P. and D. Munger. 2012. "On Figures of Merit for Randomly-Shifted Lattice Rules". In *Monte Carlo and Quasi-Monte Carlo Methods 2010*, edited by H. Woźniakowski and L. Plaskota, 133–159. Berlin: Springer-Verlag.

L'Ecuyer, P. and D. Munger. 2016. "Algorithm 958: Lattice Builder: A General Software Tool for Constructing Rank-1 Lattice Rules". *ACM Transactions on Mathematical Software* 42(2):Article 15.

L'Ecuyer, P., M. Nakayama, A. B. Owen, and B. Tuffin. 2023. "Confidence Intervals for Randomized Quasi-Monte Carlo Estimators". In *2023 Winter Simulation Conference (WSC)*, 445–456 https://dl.acm.org/doi/10.5555/3643142.3643179.

Lemieux, C. and P. L'Ecuyer. 2003. "Randomized Polynomial Lattice Rules for Multivariate Integration and Simulation". *SIAM Journal on Scientific Computing* 24(5):1768–1789.

Marion, P., M. Godin, and P. L'Ecuyer. 2020. "An algorithm to compute the t-value of a digital net and of its projections". *Journal of Computational and Applied Mathematics* 371(June):112669.

Matousěk, J. 1998. "On the L_2-discrepancy for Anchored Boxes". *Journal of Complexity* 14:527–556.

Matoušek, J. 1999. *Geometric Discrepancy: An Illustrated Guide*. Berlin: Springer-Verlag.

Matsumoto, M., M. Saito, and K. Matoba. 2014. "A Computable Figure of Merit for Quasi-Monte Carlo Point Sets". *Mathematics and Computers in Simulation* 83(287):1233–1250.

Matsumoto, M. and T. Yoshiki. 2013. "Existence of fast convergent quasi-Monte Carlo point sets via Walsh figure of merit". In *Monte Carlo and Quasi-Monte Carlo Methods 2012*, edited by J. Dick, F. Y. Kuo, G. W. Peters, and I. H. Sloan, 569–579. Heidelberg: Springer.

Niederreiter, H. 1992. *Random Number Generation and Quasi-Monte Carlo Methods*, Volume 63 of *SIAM CBMS-NSF Reg. Conf. Series in Applied Mathematics*. SIAM.

Owen, A. B. 1995. "Randomly Permuted (t,m,s)-Nets and (t,s)-Sequences". In *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, edited by H. Niederreiter and P. J.-S. Shiue, Volume 106 of *Lecture Notes in Statistics*, 299–317. Springer-Verlag.

Owen, A. B. 1997. "Monte Carlo Variance of Scrambled Equidistribution Quadrature". *SIAM Journal on Numerical Analysis* 34(5):1884–1910.

Owen, A. B. 2003. "Variance with Alternative Scramblings of Digital Nets". *ACM Transactions on Modeling and Computer Simulation* 13(4):363–378.

Schmid, W. C. 2001. "Projections of digital nets and sequences". *Mathematics and Computers in Simulation* 55(1-3):239–247.

Sloan, I. H. and S. Joe. 1994. *Lattice Methods for Multiple Integration*. Oxford: Clarendon Press.

Sobol', I. M. 1967. "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.

Sobol, I. M. and D. I. Asotsky. 2003. "One More Experiment on Estimating High-dimensional Integrals by quasi-Monte Carlo Methods". *Mathematics and Computers in Simulation* 62(3-6):255–263.

Wiart, J., C. Lemieux, and G. Y. Dong. 2021. "On the dependence structure and quality of scrambled (t,m,s)-nets". *Monte Carlo Methods and Applications* 27:1–26.

Yoshiki, T. 2017. "Bounds on Walsh coefficients by dyadic difference and a new Koksma-Hlawka type inequality for quasi-Monte Carlo integration". *Hiroshima Mathematical Journal* 47:155–179.

Yue, R.-X. and F. J. Hickernell. 2002. "The Discrepancy and Gain Coefficients of Scrambled Digital Nets". *Journal of Complexity* 18(1):135–151.

AUTHOR BIOGRAPHIES

PIERRE L'ECUYER is a Professor in the Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, Canada. He is a member of the CIRRELT and GERAD. His main research interests are random number generation, quasi-Monte Carlo methods, efficiency improvement via variance reduction, sensitivity analysis and optimization of discrete-event stochastic systems. He has published over 300 scientific articles and has developed software libraries and systems for random number generation and stochastic simulation. He received the LPAA from the INFORMS Simulation Society in 2020. His email address is lecuyer@iro.umontreal.ca and his website is https://www.iro.umontreal.ca/~lecuyer.

YOUSSEF CHERKANIHASSANI is a master student in mathematical and computational finance at Université de Montréal, Canada. Email: youssef.cherkani.hassani@umontreal.ca.

MOHAMED EL AMINE DERKAOUI was a master student in mathematical and computational finance at Université de Montréal, Canada. He also has a master degree in statistics. Email: mohamed.el.amine.derkaoui@umontreal.ca.