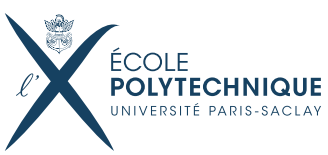
LATNET BUILDER CONSTRUCTION OF QUASI-MONTE CARLO RULES DIGITAL NETS AND INTERLACED DIGITAL NETS

Research Internship Report

Maxime Godin (X2015)

Option: Department of Applied Mathematics
Field: Probabilistic and Statistical Modelling
Referent: Vincent Bansaye
Supervisor: Pierre L'Ecuyer
Date: 25 mars 2018 - 25 août 2018
Organism: University of Montreal
Department of Computer Science and Operations Research



Abstract

Quasi-Monte Carlo methods (QMC) offer an interesting alternative to traditional *Monte-Carlo methods* (MC) for numerical integration. The integration error can indeed be much smaller when using QMC instead of MC. QMC methods require the construction of highly uniform point sets which serve as integration rule. Such point sets are called *QMC rules*. It is possible to use pre-computed QMC rules but they may not be adapted to the problem at hand. As an alternative, *LatNet Builder*, a software developed by the Stochastic Simulation and Optimization Laboratory at University of Montreal, offers to construct good QMC rules on-the-fly, depending on the function to integrate. The notion of 'goodness' is measured through quality criteria which depends on the problem. Computer searches are then used to find good point sets. During this internship, we significantly extended the software by adding two popular techniques for constructing such points: *digital nets* and *interlaced digital nets*. Specific quality criteria have been implemented for these constructions, with new efficient evaluation algorithms. The theory of digital nets, interlaced digital nets and their quality measures are discussed in the report. Numerical experiments also demonstrate the interest of using LatNet Builder to search for good integration rules.

Les *méthodes quasi-Monte Carlo* (QMC) offrent une alternative intéressante aux traditionnelles *méthodes Monte Carlo* (MC) pour des problèmes d'intégration numérique. L'erreur d'intégration peut en effet être réduite en utilisant les méthodes QMC plutôt que les méthodes MC. Les méthodes QMC utilisent des points hautement uniformes qui servent de règle d'intégration. De telles règles sont appelées *règles QMC*. Bien qu'on puisse utiliser des règles QMC tabulées, ces dernières peuvent ne pas être adaptées au problème que l'on veut résoudre. *LatNet Builder*, logiciel développé au Laboratoire de Simulation Stochastique et d'Optimisation, propose au contraire de construire de bonnes règles QMC à la volée. Celles-ci seront adaptées à la fonction à intégrer. La qualité d'une règle QMC est mesurée par des critères de qualités qui dépendent du problème. Les meilleures règles QMC sont ensuite trouvées par recherche informatique. Durant ce stage, nous avons significativement étendu LatNet Builder en y ajoutant deux techniques populaires pour la construction de règles QMC : les *réseaux digitaux* et les *réseaux digitaux entrelacés*. Des critères de qualité spécifiques ont été implémentés pour ces constructions, avec de nouveaux algorithmes d'évaluation plus efficaces en termes de temps de calcul. La théorie des réseaux digitaux et des réseaux digitaux entrelacés, ainsi que de leurs mesures de qualité, sont discutées dans ce rapport. Des expériences numériques démontrent par ailleurs l'intérêt d'utiliser LatNet Builder pour chercher de bonnes règles d'intégration.

Déclaration d'intégrité relative au plagiat

Je soussigné Maxime, Godin certifie sur l'honneur:

1. Que les résultats décrits dans ce rapport sont l'aboutissement de mon travail.
2. Que je suis l'auteur de ce rapport.
3. Que je n'ai pas utilisé des sources ou résultats tiers sans clairement les citer et les référencer selon les règles bibliographiques préconisées.

Je déclare que ce travail ne peut être suspecté de plagiat.

Le 29 août 2018, Maxime Godin

ÉCOLE
POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

Acknowledgements

My thanks first go to Professor Pierre L'Ecuyer, head of the Stochastic Simulation and Optimization Laboratory from the Department of Computer Science and Operations Research at University of Montreal. My internship under his supervision was a great opportunity to develop my research skills. Thank you Pierre for the corn and the tongue! I would also like to thank Florian Puchhammer for his support and his kindness. You always made yourself available when we needed your help. Thank you also to all the post-doctoral and PhD students from the lab, as well as to all the interns. In particular, I cannot forget to thank my dear friend and colleague Pierre Marion. His outstanding work and his unquenchable curiosity undoubtedly contributed to the success of this project. We surely learn more together than we would have alone.

Contents

1 Introduction

In stochastic simulation, a common problem is to estimate the value of an expectation. More precisely, suppose that we want to estimate the expectation $\mu = \mathbb{E}[X]$ of a random variable X over a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. We assume that we can simulate the random variable X using s uniform random variables over $[0,1)$, i.e. that there exists a function $f : [0,1)^s \to \mathbb{R}$, so that for any random variable $\boldsymbol{U}$ with uniform distribution on $[0,1)^s$, X and $f(\boldsymbol{U})$ have the same distribution. Then we have

$$\mu = \mathbb{E}\left[f(\boldsymbol{U})\right] = \int_{[0,1)^s} f(\boldsymbol{u})\,\mathrm{d}\boldsymbol{u}.$$

Monte Carlo methods (MC) offer a very general solution to this problem. If $(\boldsymbol{U}_i)_{i\in\mathbb{N}}$ is a sequence of independent random variables with uniform distribution on $[0,1)^s$, the strong law of large numbers guarantees the almost sure convergence of the *empirical mean* $\hat{\mu}_n$ towards μ:

$$\hat{\mu}_n = \frac{1}{n}\sum_{i=0}^{n-1} f(\boldsymbol{U}_i) \xrightarrow[n\to+\infty]{a.s.} \mathbb{E}\left[f(\boldsymbol{U})\right] = \mu.$$

Moreover, if X is square-integrable ($\mathbb{E}\left[|X|^2\right] < +\infty$), the central limit theorem provides a confidence interval whose length typically converges towards 0 at a $\mathcal{O}(n^{-1/2})$ rate.

Quasi-Monte Carlo methods (QMC) use instead a deterministic approach. The independent random points $\boldsymbol{U}_i$ are replaced by a set of deterministic points $\mathcal{P}_n = \{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{n-1}\} \subset [0,1)^s$. $\mathcal{P}_n$ is called a *quasi-Monte Carlo rule*. QMC methods estimates μ by

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

We typically want some guarantees on the integration error for a certain class of functions. For some Banach function spaces $\mathcal{W}$ and point sets $\mathcal{P}_n$, the *worst-case error*, also called the *discrepancy*

$$e(\mathcal{W}, \mathcal{P}_n) = \sup_{f\in\mathcal{W}, \|f\|_{\mathcal{W}}\leq 1} \left| \int_{[0,1)^s} f(\boldsymbol{u})\,\mathrm{d}\boldsymbol{u} - \frac{1}{n}\sum_{i=0}^{n-1} f(\boldsymbol{u}_i) \right|, \tag{1}$$

can be computed efficiently or bounded by an easily computable quantity.

The efficiency of QMC methods, comparatively with MC methods, relies on the existence of *low-discrepancy point sets* whose discrepancy converges towards 0 faster than $\mathcal{O}(n^{-1/2})$, which is the typical rate for independent uniform random points. Moreover, QMC methods provide a worst-case error bound instead of just a confidence interval.

Randomized quasi-Monte Carlo methods (RQMC) reintroduce randomness in the deterministic approach of QMC methods. More concretely, a deterministic low discrepancy point set $\mathcal{P}_n = \{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{n-1}\}$ is randomized according to a specific procedure, yielding a random point set $\{\boldsymbol{U}_0, \ldots, \boldsymbol{U}_{n-1}\}$. This randomization technique is designed in a way such that each $\boldsymbol{U}_i$ is uniform over $[0,1)^s$ and that the random point set as a whole is a low-discrepancy point set. Then μ is estimated by

$$\hat{\mu}_{n,\,rmqc} = \frac{1}{n}\sum_{i=0}^{n-1} f(\boldsymbol{U}_i).$$

The low-discrepancy property implies the convergence of $\hat{\mu}_{n,\,rmqc}$ towards μ. For some randomizations, Banach function spaces $\mathcal{W}$ and point sets $\mathcal{P}_n$, the *mean square worst-case error*

$$\hat{e}^2(\mathcal{W}, \mathcal{P}_n) = \mathbb{E}\left[\sup_{f\in\mathcal{W}, ||f||_{\mathcal{W}}\leq 1} \left| \int_{[0,1)^s} f(\boldsymbol{u})\,\mathrm{d}\boldsymbol{u} - \frac{1}{n}\sum_{i=0}^{n-1} f(\boldsymbol{U}_i)\right|^2\right] \tag{2}$$

can be computed efficiently. When X is square-integrable , this yields an upper bound on the variance

$$\mathrm{Var}\left[\hat{\mu}_{n,\,rmqc}\right] \leq ||f||^2_{\mathcal{W}}\ \hat{e}^2(\mathcal{W}, \mathcal{P}_n)$$

which may converges towards zero faster than the squared worst-case error $e^2(\mathcal{W}, \mathcal{P}_n)$ of the corresponding deterministic QMC method. This gives another important justification for randomization: the variance of the estimator may converge towards 0 at a faster rate than the variance of the MC estimator.

The Stochastic Simulation and Optimization Laboratory of the Department of Computer Science and Operations Research, led by Pr. Pierre L'Ecuyer, develops *LatNet Builder*, an open-source software which constructs good QMC rules, i.e. point sets for which the integration error (worst-case error for QMC methods or mean square worst-case error for RQMC methods), is small.

Given a class of functions, LatNet Builder uses computer searches to find good QMC rules. The search space is in practice huge and therefore, a key issue is to find quality criteria which can be easily computed. During this internship, our main contribution was to extend LatNet Builder to two classes of QMC rules, *digital nets* and *interlaced digital nets*. This work first involved reviewing the literature to understand the general framework of QMC and RQMC methods. Then, we determined which quality criteria were adapted for the point sets at hand. We also had to design efficient evaluation algorithms for these quality criteria. Finally, we implemented the functionalities in the software and released a new version with an updated documentation.

By developing this software, the team aims at promoting the use of QMC and RQMC methods for stochastic simulation and numerical integration. These methods have proven useful to improve the performance of numerical computations and the scientific community could benefit from having an easier access to them. To our knowledge, LatNet Builder is currently the most complete open-source software available for constructing good QMC rules.

This report is organized as follows. In section 2 we give the definition of digital nets and interlaced digital nets. Then in section 3, we explain the idea behind searching for 'good' digital nets. In particular, we explicit the search spaces and the quality measures which can be used in LatNet Builder for digital nets. In section 4, we focus on the software by summarising our main contributions and providing a numerical experiment showing the interest of searching for good QMC rules. We also present the efficient evaluation algorithms we designed for the quality measures presented in this report.

2 Digital nets

In this section, we introduce the *digital net* framework which provides an explicit construction of QMC rules. This class of point sets is the first one we added to the software. In LatNet Builder, we made the choice to restrict ourself to base 2. This choice is meant as a convenience for software implementation as computers work in binary. It also allows to fasten computations as the representation of data is very close to hardware. The construction of digital nets relies on linear algebra in the finite field $\mathbb{F}_2$. For each coordinate, a generating matrix is used to produce the coordinate values of the point set. We first present the definition of digital nets with some classical constructions of generating matrices. Then we introduce a second class of point sets called interlaced digital nets.

2.1 Definition and classical constructions

The following definition is taken from [5] in the special case of base 2.

Definition 2.1 (Digital net)**.** Let $\mathbb{F}_2$ be the finite field with 2 elements (denoted 0 and 1). For a given dimension $s \geq 1$ and positive integers k and r with $r \geq k$, let $M_1, \ldots, M_s$ be $r \times k$ matrices over $\mathbb{F}_2$.

The *digital net with generating matrices* $M_1, \ldots, M_s$ is the point set $\mathcal{P}_{2^k} = \{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{2^k-1}\}$ in $[0,1)^s$ with 2^k points such that for $0 \leq i < 2^k$, $\boldsymbol{u}_i = (u_{i,1}, \ldots, u_{i,s})^T$. The components of $\boldsymbol{u}_i$ are defined by

$$u_{i,l} = \sum_{j=1}^{r} t_{i,l,j} 2^{-j}.$$

for $l = 1, \ldots, s$ where $\boldsymbol{t}_l = M_l \times \boldsymbol{i} = (t_{i,l,1}, \ldots, t_{i,l,r})^T$ and $\boldsymbol{i} = (a_0, \ldots, a_{k-1})^T$ denotes the digits in $\mathbb{F}_2$ of the 2-adic expansion $i = \sum_{j=0}^{k-1} a_j 2^j$.

Digital nets can be much more uniformly distributed then random points. For now we will just give a visual illustration of this phenomenon. Figure 1 compares independent and uniformly distributed random points with a digital net in $[0,1)^2$ for $n = 2^{10}$ points. One can clearly see 'holes' on Figure 1a which do not appear for the digital net on Figure 1b. The presence of 'holes' motivates the search of more evenly distributed point sets such as digital nets.

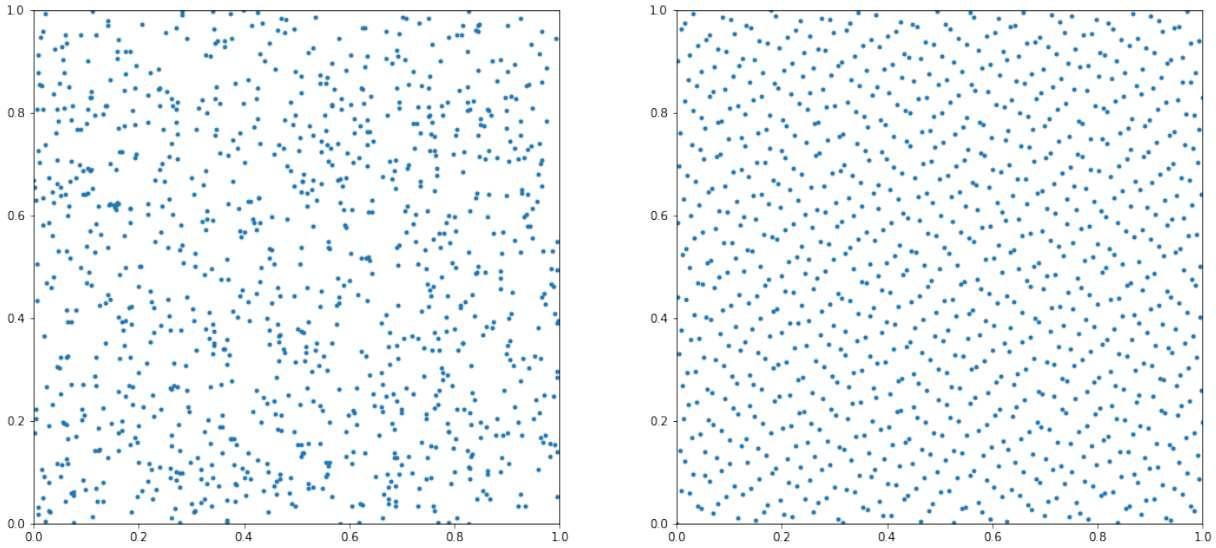
2.1.1 Sobol' construction

A popular technique for constructing digital nets is the *Sobol' construction*. These construction was first proposed by Sobol' in [20]. The following definition is from [9].

Definition 2.2 (Sobol' net)**.** For a given dimension $s \geq 1$, for $1 \leq j \leq s$, let

$$Q_j(z) = z^{e_j} + a_{1,j} z^{e_j - 1} + \cdots + a_{e_j - 1, j} z + 1$$

be a primitive polynomial of degree e_j over $\mathbb{F}_2$.



(a) uniform random point set (b) digital net

Figure 1: Comparison between a random uniform point set and a digital net for $n = 2^{10}$ points in $[0, 1)^s$

Define the sequence of positive integers $\{m_{1,j}, m_{2,j}, \dots\}$ by the recurrence relation

$$m_{r,j} = 2a_{1,j}m_{r-1,j} \oplus 2^2 a_{2,j}m_{r-2,j} \oplus \cdots \oplus 2^{e_j-1}a_{e_j-1,j}m_{r-e_j\oplus 1,j} \oplus 2^{e_j}m_{r-e_j,j} \oplus m_{r-e_j,j}$$

where the initial values $m_{1,j}, \dots, m_{e_j,j}$ can be chosen freely provided that each $m_{r,j}$, $1 \leq r \leq e_j$ is odd and less than 2^r and $\oplus$ denotes the bitwise exclusive-or operator.

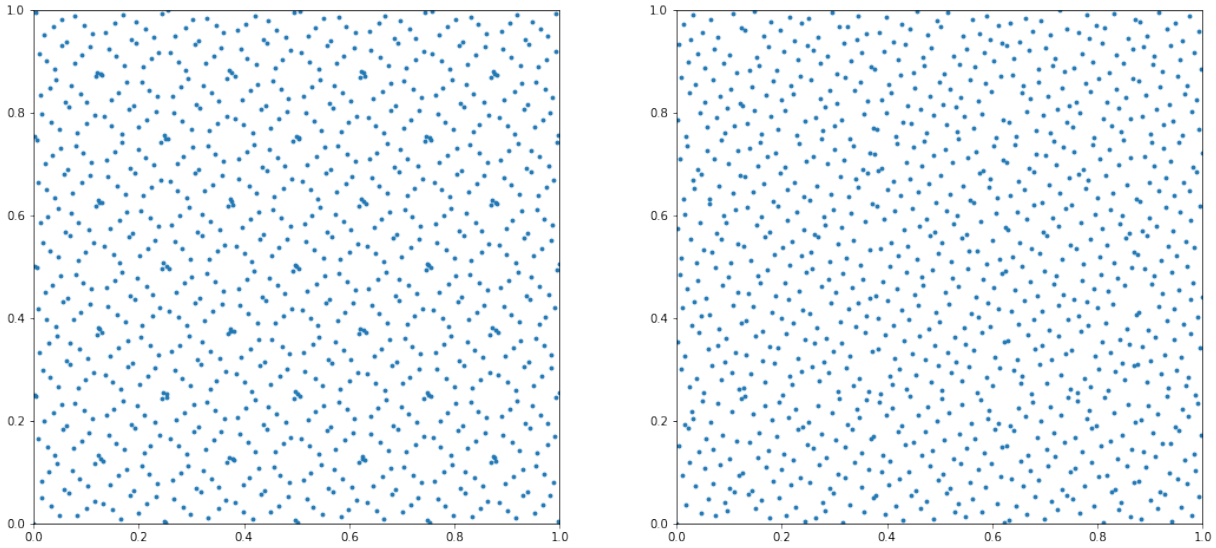
The $\{m_{1,j}, m_{2,j}, \dots\}$ are called the *direction numbers* for coordinate j. The first coordinate $j = 1$ is a special case for which all the $m_{r,1}$ equal 1. Write $m_{r,j} = \sum_{l=1} m_{r,j,l}2^{r-l}$ the 2-adic expansion of $m_{r,j}$. Then, for any positive integer k and for $1 \leq j \leq s$, define the matrix of size $k \times k$ over $\mathbb{F}_2$

$$M_j = \begin{bmatrix} 1 & m_{2,j,1} & m_{3,j,1} & \cdots & m_{k,j,1} \\ 0 & 1 & m_{3,j,1} & \cdots & m_{k,j,2} \\ 0 & 0 & 1 & \cdots & m_{k,j,3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}.$$

The digital net defined by these matrices is called the *Sobol' net with direction numbers* $m_{r,j}$, $1 \leq j \leq s$, $1 \leq r \leq e_j$.

2.1.2 Polynomial lattice rules

Another popular technique to construct digital net are *polynomial lattice rules*. They were first introduced by Niederreiter in [18]. Denote by $\mathbb{F}_2[z]$ the ring of polynomials over $\mathbb{F}_2$ and by $\mathbb{L}_2[z]$ the field of formal Laurent series. We have the following definition from [5].



(a) Sobol' net with direction numbers $(1), (1)$

(b) polynomial lattice rules with modulus $Q = 1 + z^3 + +z^{10}$ and generating vector $\boldsymbol{a} = (1,\ z^5 + z^8 + z^9)$

Figure 2: Comparison between a Sobol' net and a polynomial lattice rule for $n = 2^{10}$ points in $[0, 1)^s$

Definition 2.3 (Polynomial lattice rule)**.** Let Q be a polynomial in $\mathbb{F}_2[z]$ of degree k and $\boldsymbol{a} = (a_1, \ldots, a_s)$ be a vector of s polynomials $\mathbb{F}_2[z]$. Suppose for $1 \leq j \leq s$ that $\deg a_j \leq k$ and consider the formal expansion

$$\frac{a_j(z)}{Q(z)} = \sum_{l=w_j}^{+\infty} u_l^{(j)} z^{-l} \in \mathbb{L}_2[z]$$

where $w_j \leq 1$. Define the $k \times k$ matrices $M_1, \ldots, M_s$ over $\mathbb{F}_2$ where the elements $m_{l,r+1}^{(j)}$ of the matrix M_j are given by

$$m_{l,r+1}^{(j)} = u_{r+l}^{(j)} \in \mathbb{F}_2,$$

for $1 \leq j \leq s$, $1 \leq l \leq k$ and $0 \leq r \leq k-1$. Then the *polynomial lattice rule with modulus* Q *and generating vector* $\boldsymbol{a} = (a_1, \ldots, a_s)$ is the digital net with generating matrices $M_1, \ldots, M_s$.

Figure 2 shows two examples of digital net constructed using the two previous techniques: a Sobol' net and a polynomial lattice rule.

2.2 Interlaced digital nets

Interlaced digital nets aim at improving the rate of convergence of the integration error of smooth functions. The idea of interlacing digital nets was first proposed by Dick in [6]. The construction principle of *interlaced digital nets* in dimension s is based on a *digit interlacing function* applied to classical (non-interlaced) digital nets whose number of components is $d \times s$

where d is an integer greater than 1 called the *interlacing factor*. More precisely, in this approach, a digital net in dimension $d \times s$ is transformed (or interlaced) into a point set in dimension s by applying the digit interlacing function to each of its d consecutive components. In what follows, we provide a definition of interlaced digital nets following [8].

Definition 2.4 (Digit interlacing function)**.** The *digit interlacing function with interlacing factor* d where d in an integer greater than 1 is the function $\mathcal{E}_d : [0,1)^d \to [0,1)$ defined by

$$\mathcal{E}_d(z_1, \dots, z_d) = \sum_{a=1}^{\infty} \sum_{r=1}^{d} z_{r,a} 2^{-r-(a-1)d}$$

where the 2-adic expansion of $z_l \in [0,1)$ is denoted by $z_l = z_{l,1}2^{-1} + z_{l,2}2^{-2} + \dots$ for $1 \le l \le d$ and where it is assumed that the expansion is unique in the sense that infinitely many digits are different from 1.

Definition 2.5 (Interlaced digital net)**.** The interlaced digital net $\mathcal{P}_{2^k} = \{\boldsymbol{u}_0, \dots, \boldsymbol{u}_{2^k-1}\}$ with 2^k points in $[0,1)^s$ with interlacing factor d and underlying digital net $\bar{\mathcal{P}}_{2^k} = \{\boldsymbol{z}_0, \dots, \boldsymbol{z}_{2^k-1}\}$ in $[0,1)^{ds}$ is the point set defined by

$$\boldsymbol{u}_i = \left(\mathcal{E}_d(z_{i,1}, \dots, z_{i,d}), \dots, \mathcal{E}_d(z_{i,(s-1)d+1}, \dots, z_{i,sd})\right).$$

The digit interlacing function is designed in a way that an interlaced digital net is also a classical digital net. However, its generating matrices are not square as it is the case for Sobol' nets and polynomial lattice rules. More precisely, we have the following proposition whose proof immediately follows from Definition 2.5.

Proposition 2.6. *Let $\mathcal{P}_{2^k}$ be an interlaced digital net with 2^k points in $[0,1)^s$ with interlacing factor d and underlying digital net $\bar{\mathcal{P}}_{2^k}$. Denote by $\bar{M}_1, \dots, \bar{M}_{ds}$ the generating matrices of $\bar{\mathcal{P}}_{2^k}$. For $j = 1, \dots, s$, let M_j be the $kd \times k$ matrix whose l-th row is the $\lceil l/d \rceil$-th row of $\bar{M}_{((l-1) \bmod d)+1}$. Then $\mathcal{P}_{2^k}$ is the digital net with generating matrices $M_1, \dots, M_s$.*

Remark 2.7. Note that using interlaced digital nets, the precision on the points is multiplied by d compared to square generating matrices. Using Sobol' nets or polynomial lattice rules, the points only have a k-bit precision where 2^k is the number of points.

Figure 3 show an interlaced digital net with $n = 2^{10}$ points in $[0,1)^s$ with interlacing factor 2.

3 Searching for good digital nets

As stated in the introduction, LatNet Builder uses computer searches to find good QMC rules. For this purpose, we need a search space, a quality criteria and an exploration method. These three ingredients are at the core of LatNet Builder. For each one, numerous possibilities are supported. Thus, many different searches can be done by the software.

Quality criteria are designed in a way that point sets with small merit values have a small integration error. Finding such point sets is therefore a way to improve the performance of QMC and RQMC methods.

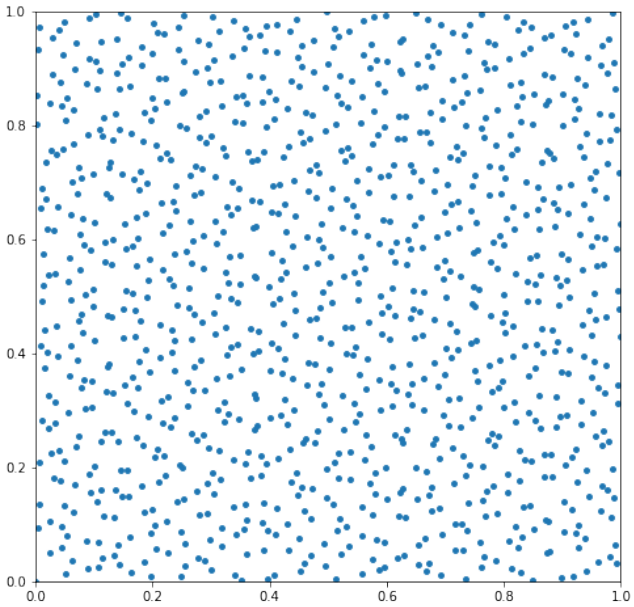


Figure 3: Interlaced digital net with $n = 2^{10}$ points in $[0,1)^2$ with interlacing factor $d = 2$ whose underlying digital net is the polynomial lattice rule with modulus $Q = 1 + z^3 + z^{10}$ and generating vector $\boldsymbol{a} = (1,\ z^5 + z^8 + z^9,\ 1 + z + z^2 + z^6 + z^8 + z^9,\ 1 + z + z^4 + z^6 + z^7 + z^8 + z^9)$

3.1 Search spaces

As a digital net is uniquely defined by its generating matrices, search spaces are actually sets of s-dimensional vectors of matrices in $\mathbb{F}_2$. The basic search space is defined as the s-ary cartesian power of the set of invertible generating matrices. We only consider invertible matrices as digital nets with singular generating matrices do not exhibit the permutation property (Proposition 4.3). More formally, for a positive integer k, let $GL_k(\mathbb{F}_2)$ be the set of $k \times k$ invertible matrices. Then the basic search space for digital nets with 2^k points in $[0,1)^s$ is

$$\mathcal{S}_{k,s} = \left(GL_k(\mathbb{F}_2)\right)^s.$$

This space is often huge as the cardinality of $GL_k(\mathbb{F}_2)$ is bounded from below [17] by

$$|GL_k(\mathbb{F}_2)| = \prod_{m=0}^{k-1} \left(2^k - 2^m\right) \geq 0.28 \times 2^{k^2}.$$

It is however possible to restrict the search space to subsets of $\mathcal{S}_{k,s}$. This is one of the reasons to use the Sobol' and polynomial lattice rules constructions presented in section 2.1.

3.1.1 Sobol' construction

For the Sobol' construction we have the search space

$$\mathcal{S}_{k,s}^{sob} = \prod_{j=1}^{s} \mathcal{S}_k^{sob,(j)}$$

where $\mathcal{S}_{k,s}^{sob,(j)}$ is the set of generating matrices associated to the primitive polynomial Q_j. Furthermore, as a generating matrix for coordinate j is uniquely defined by its e_j direction numbers, we have that

$$|\mathcal{S}_k^{sob,(j)}| = 2^{\sum_{l=1}^{\min(e_j,k)} 2^{l-1}}.$$

Instead of searching for generating matrices, we directly search for those direction numbers.

3.1.2 Polynomial lattice rules

For polynomial lattice rules with modulus Q of degree k

$$\mathcal{S}_{Q,s}^{poly} = \left(\mathcal{S}_Q^{poly}\right)^s$$

where S_Q^{poly} is the set of generating matrices constructed as explained in Definition 2.3 with a_j constrained to be coprime with Q for $j = 1, \dots, s$. This latest constraint on a_j forces the resulting matrix to be invertible ([5, Remark 10.3]). As a generating matrix is uniquely defined by its generating value a_j, we have that

$$|\mathcal{S}_Q^{poly)}| = \phi(Q)$$

where $\phi(Q)$ denotes the Euler's totient function which counts the number of polynomials of degree lower than k which are coprime with Q. As for the Sobol' construction, we do not search the generating matrices themselves but rather the generating vector $\boldsymbol{a}$.

All in all, LatNet Builder offer three search spaces for digital nets in $[0,1)^s$ with 2^k points: the set of s-dimensional vector of $k \times k$ invertible matrices $\mathcal{S}_{s,k}$, the Sobol' nets $\mathcal{S}_{s,k}^{sob}$ and the polynomial lattice rules $\mathcal{S}_{s,Q}^{poly}$ for a given modulus Q of degree k. The last two spaces are included in the first one and much smaller in practice.

Remark 3.1. For interlaced digital nets with 2^k points in $[0,1)^s$ with interlacing factor s, we do not search for the point set itself but rather for the underlying digital net. Therefore, the search space can be any of the above spaces with a dimension taken equal to $d \times s$.

3.2 Quality criteria

In this section we present various quality criteria for digital nets. They are also called quality measures or figures of merit. Ideally, good quality criteria would either be the worst-case error for QMC methods or the mean square worst-case error for RQMC methods for some function space $\mathcal{W}$. As closed and efficient formulas for these errors are not always available, it may be necessary to use instead upper bounds. In this case, we do not directly minimize the error. If the bound is sufficiently tight, the result may still be satisfactory. In any case, we still have a bound on the error even if we are not directly minimizing it.

For sake of brevity, we only present three families of figures of merit. LatNet Builder implements much more quality criteria. As explained in section 4.3, from the computational

view point, these figures of merit are evaluated using the same algorithms. Other figures of merit which require different evaluation algorithms are detailed in the report of Pierre Marion.

Before going into details, we need to introduce the concept of weighted function spaces. The following figures of merit are indeed derived in such weighted spaces. Given a dimension s, we call general or projection-dependent weights any collection of positive real numbers $\boldsymbol{\gamma} = \{\gamma_{\mathfrak{u}} : \mathfrak{u} \subseteq \{1, \dots, s\}\}$. The idea behind weights is to decompose the function f to integrate as a sum of lower dimensional functions.

Definition 3.2 (ANOVA decomposition)**.** A function $f \in L^2([0,1)^s)$ verifies the ANOVA decomposition

$$f(\boldsymbol{y}) = \sum_{\mathfrak{u} \subseteq \{1,\dots,s\}} f_{\mathfrak{u}}(\boldsymbol{y}_{\mathfrak{u}})$$

with

$$f_{\mathfrak{u}}(\boldsymbol{y}_{\mathfrak{u}}) = \int_{[0,1]^{s-|\mathfrak{u}|}} f(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{y}_{-\mathfrak{u}} - \sum_{\mathfrak{v} \subseteq \mathfrak{u}} f_{\mathfrak{v}}(\boldsymbol{y}_{\mathfrak{v}})$$

where $\boldsymbol{y}_{-\mathfrak{u}}$ stands for all the components of the s-dimensional vector $\boldsymbol{y}$ which are not in $\boldsymbol{y}_{\mathfrak{u}}$. The ANOVA components verify the orthogonality property

$$\int_{[0,1]^s} f_{\mathfrak{u}}(\boldsymbol{y}_{\mathfrak{u}}) f_{\mathfrak{v}}(\boldsymbol{y}_{\mathfrak{v}}) \, \mathrm{d}\boldsymbol{y} = 0 \text{ for } \mathfrak{u} \neq \mathfrak{v}.$$

The ANOVA decomposition may be used to decompose the variance of a function. If we denote by the variance of f as $\mathrm{Var}(f) = \int_{[0,1]^s} |f(\boldsymbol{y})|^2 \, \mathrm{d}\boldsymbol{y} - \left(\int_{[0,1]^s} f(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{y}\right)^2$, then we have from the orthogonality property

$$\mathrm{Var}(f) = \sum_{\mathfrak{u} \in \{1,\dots,s\}} \mathrm{Var}(f_{\mathfrak{u}}).$$

Each lower dimensional function $f_{\mathfrak{u}}$ may more or less hard to integrate. This means that the QMC rule used to integrate the function should somehow focus on its most difficult components. More precisely, weights emphasize the importance of some sets of coordinates, also called projections. The more important the projection, the greater the weight. All the figures of merit presented in this report can be expressed as a weighted ℓ^q norm with respect to the projections $\mathcal{P}_n(\mathfrak{u})$ of $\mathcal{P}_n$. Such figures of merit are called *weighted figures of merit* [16].

Definition 3.3 (Weighted figure of merit)**.** A *weighted figure of merit* is a figure of merit which can be written

$$[\mathcal{D}_q(\mathcal{P}_n)]^q = \sum_{\emptyset \neq \mathfrak{u} \subseteq \{1,\dots,s\}} \gamma_{\mathfrak{u}}^q [\mathcal{D}_{\mathfrak{u}}(\mathcal{P}_n)]^q, \tag{3}$$

where $1 \leq q < +\infty$ is a number called the *norm-type* and where, for every projection $\mathfrak{u}$, the *projection-dependent weight* $\gamma_{\mathfrak{u}}^q$ is a positive constant and the *projection-dependent figure of merit* $\mathcal{D}_{\mathfrak{u}}(\mathcal{P}_n)$ depends only on $\mathcal{P}_n(\mathfrak{u})$.

The projection-dependent figure of merit $\mathcal{D}_{\mathfrak{u}}(\mathcal{P}_n)$ accounts for the quality of projection $\mathfrak{u}$. If $\gamma_{\mathfrak{u}}$ is large, a point set with a bad (large) projection-dependent figure of merit will be penalized. Weighted function spaces are normed function spaces whose norm accounts for the differences between the projections in the 'difficulty scale' by using weights. Examples of such spaces will be provided in the following sections.

3.2.1 Bound on the worst-case error in the weighted anchored Sobolev space

In this section, we first present a theoretical result on the worst-case error in the *weighted anchored Sobolev space*. We then use a bound on the *weighted star discrepancy* to derive a quality measure for digital nets.

Worst-case error in the weighted unanchored Sobolev space We now provide the definition of the function space in which we want to derive the worst-case error. The following definition of the weighted anchored Sobolev space is taken from [11].

Definition 3.4 (Weighted anchored Sobolev space)**.** For $s \in \mathbb{N}^*$, $1 \leq q \leq +\infty$ and positive weights $\boldsymbol{\gamma}$, the *weighted anchored Sobolev space* $\mathcal{W}_{s,\boldsymbol{\gamma}}^q$ is defined as the subset of functions over $[0,1)^s$ in L^q with mixed first derivatives in L^q. For $1 \leq r \leq +\infty$, $\mathcal{W}_{s,\boldsymbol{\gamma}}^q$ can be equipped with the norm

$$||f||_{\mathcal{W}_{s,\boldsymbol{\gamma}}^{q,r}} = \left(\sum_{\mathrm{u} \subseteq \{1,\dots,s\}} \left\| \gamma_{\mathrm{u}}^{-1/2} \frac{\partial^{|\mathrm{u}|}}{\partial z_{\mathrm{u}}} f(z_{\mathrm{u}}, 1) \right\|_{L^q}^r \right)^{1/r}$$

where the L^q norm of a function h is defined by

$$||h||_{L^q} = \begin{cases} \left(\int_{[0,1]^s} |h(\boldsymbol{y})|^q \, \mathrm{d}\boldsymbol{y} \right)^{1/q} & \text{for } 1 \leq q < +\infty, \\ \operatorname*{ess\,sup}_{\boldsymbol{y} \in [0,1]^s} |f(\boldsymbol{y})| & \text{for } q = +\infty. \end{cases}$$

The resulting normed vector space, denoted $\mathcal{W}_{s,\boldsymbol{\gamma}}^{q,r}$, is a Banach space.

Remark 3.5. Here, for sake of simplicity, the anchor of the space $W_{s,\boldsymbol{\gamma}}^{q,r}$ is 1 but other choices of anchor could be made.

The worst-case error in $W_{s,\boldsymbol{\gamma}}^{q,r}$ for a point set $\mathcal{P}_n$ can be expressed through the *local discrepancy of* $\mathcal{P}_n$ [5, Definition 2.13]).

Definition 3.6 (Local discrepancy)**.** For a point set $\mathcal{P}_n = \{\boldsymbol{u}_0, \dots, \boldsymbol{u}_{n-1}\}$ consisting of n points in $[0,1)^s$ the function $\Delta_{\mathcal{P}_n} : [0,1]^s \to \mathbb{R}$, given by

$$\Delta_{\mathcal{P}_n}(\boldsymbol{y}) = \frac{1}{n} \sum_{i=0}^{n-1} \mathbb{1}_{\boldsymbol{u}_i \in [\boldsymbol{0}, \boldsymbol{y}]} - \prod_{j=1}^{s} y_i,$$

is called the *local discrepancy of* $\mathcal{P}_n$.

The local discrepancy measures the difference between the proportion of points of $\mathcal{P}_n$ in the interval $[0, \boldsymbol{y})$ and its Lebesgue measure. It can be interpreted as a local measure for the deviation from the uniform distribution.

Now we can state the following theorem derived from [11, Lemma 2] which provides the worst-case error in the weighted anchored Sobolev space $W_{s,\boldsymbol{\gamma}}^{q,r}$.

Theorem 3.7 (Worst-case error in $\mathcal{W}_{s,\boldsymbol{\gamma}}^{q,r}$)**.** *The worst-case error in the Sobolev space $\mathcal{W}_{s,\boldsymbol{\gamma}}^{q,r}$ for a QMC rule $\mathcal{P}_n$ with n points is given by*

$$e(\mathcal{W}_{s,\boldsymbol{\gamma}}^{q,r}, \mathcal{P}_n) = \begin{cases} \left(\sum_{\emptyset \neq \mathfrak{u} \subseteq \{1,\dots,s\}} \left\| \gamma_{\mathfrak{u}}^{1/2} \Delta_{\mathcal{P}_n}(\boldsymbol{y}_{\mathfrak{u}}, \mathbf{1}) \right\|_{L_{q'}}^{r'} \right)^{1/r'} & \text{for } 1 \le r' < +\infty, \\ \max_{\emptyset \neq \mathfrak{u} \subseteq \{1,\dots,s\}} \left\| \gamma_{\mathfrak{u}}^{1/2} \Delta_{\mathcal{P}_n}(\boldsymbol{y}_{\mathfrak{u}}, \mathbf{1}) \right\|_{L_{q'}} & \text{for } r' = +\infty \end{cases}$$

where q' and r' are the Hölder conjugates of q and r ($1/q + 1/q' = 1$ and $1/r + 1/r' = 1$) respectively and $(\boldsymbol{y}_{\mathfrak{u}}, \mathbf{1})$ denotes the s-dimensional vector whose i-th component is y_i if $i \in \mathfrak{u}$ and 1 otherwise.

Koksma-Hlawka inequality and weighted star discrepancy In the case $q = r = 1$, Theorem 3.7 yields the *Koskma-Hlawka inequality* (Corollary 3.9) and the associated worst-case error is called the *weighted star discrepancy* (Definition 3.8).

Definition 3.8 (Weighted star discrepancy)**.** For a point set $\mathcal{P}_n$ with n points in $[0,1)^s$, the quantity

$$\mathcal{D}_{\boldsymbol{\gamma}}^*(\mathcal{P}_n) = \max_{\emptyset \neq \mathfrak{u} \subseteq \{1,\dots,s\}} \operatorname*{ess\,sup}_{\boldsymbol{y} \in [0,1]^s} \gamma_{\mathfrak{u}}^{1/2} \Delta_{\mathcal{P}_n}(\boldsymbol{y}_{\mathfrak{u}}, \mathbf{1})$$

is called the *weighted star discrepancy of $\mathcal{P}_n$*

Corollary 3.9 (Koskma-Hlawka inequality)**.** *For a function $f \in W_{s,\gamma}^{1,1}$ and a QMC rule $\mathcal{P}_n$ in $[0,1)^s$,*

$$\left| \frac{1}{n} \sum_{i=0}^{n-1} f(\boldsymbol{u}_i) - \int_{[0,1)^s} f(\boldsymbol{u}) \, \mathrm{d}\boldsymbol{u} \right| \le ||f||_{\mathcal{W}_{s,\gamma}^{1,1}} \mathcal{D}_{\boldsymbol{\gamma}}^*(\mathcal{P}_n).$$

Computing the star discrepancy for an arbitrary point set is a NP-hard problem [7]. Hence, the computation of the quality criteria given by Corollary 3.9 is not feasible in practice. However, for digital nets, it is possible to bound the weighted star discrepancy. A slight generalization of [5, Equations (5.13) and (10.4)] to general weights and digital nets indeed yields the following theorem.

Theorem 3.10 (Bound on the weighted star discrepancy)**.** *Let $\mathcal{P}_{2^k}$ be a digital net with 2^k points in $[0,1)^s$. Then, for positive weights $\boldsymbol{\gamma}$, the weighted star discrepancy $\mathcal{D}_{\boldsymbol{\gamma}}^*(\mathcal{P}_{2^k})$ is bounded by*

$$R_{\boldsymbol{\gamma}}(\mathcal{P}_{2^k}) - \sum_{\emptyset \neq \mathfrak{u} \subseteq \{1,\dots,s\}} \gamma_{\mathfrak{u}} \left(1 - \frac{1}{2^k}\right)^{|\mathfrak{u}|}$$

where

$$R_{\boldsymbol{\gamma}}(\mathcal{P}_{2^k}) = \frac{1}{2^k} \sum_{i=0}^{2^k-1} \sum_{\emptyset \neq \mathfrak{u} \subseteq \{1,\dots,s\}} \gamma_{\mathfrak{u}} \prod_{j \in \mathfrak{u}} \phi_k(u_{i,j})$$

where

$$\phi_k(u) = \begin{cases} \frac{1}{2} i_0 & \text{if } u = \sum_{i=i_0}^{\infty} \xi_i 2^i,\ 1 \le i_0 \le k,\ \xi_{i_0} \neq 0, \\ 1 + \frac{1}{2}k & \text{otherwise.} \end{cases}$$

3.2.2 Digital shift mean square worst-case error in the weighted unanchored Sobolev space

In what follows, we introduce the *digital shift randomization*. This randomization operates an exclusive-or on the binary digits of point sets, yielding random points with uniform distribution on $[0,1)^s$ and thus providing unbiased estimators for the mean of integrable functions. For this randomization, the mean square worst-case error in the *weighted unanchored Sobolev space* for a (digitally-shifted) digital net has an explicit expression. Recall that the mean square worst-case error provides a bound on the variance of the RQMC method associated to the point set. This bound is tight in the sense that there exists a function for which the variance is arbitrarily close to the bound. The mean square worst-case error can therefore be used as a figure of merit to find good digital nets.

Definition 3.11 (Digital shift)**.** Let $P_n = \{\boldsymbol{u}_0, \dots, \boldsymbol{u}_{n-1}\}$ be a point set with n points in $[0,1)^s$ and a random variable $\boldsymbol{V} = (V_1, \dots, V_s)$ with uniform distribution on $[0,1)^s$. For $i = 0, \dots, n-1$ define the random variable

$$\boldsymbol{U}_i = (u_{i,1} \oplus V_i, \dots, u_{i,s} \oplus V_s)$$

where $\oplus$ denotes the bitwise exclusive-or operator. Then the point set $\{\boldsymbol{U}_0, \dots, \boldsymbol{U}_n\}$ is the *digitally-shifted random point set associated to* $\mathcal{P}_n$.

We have the following proposition which directly derives from the previous definition [12].

Proposition 3.12. *Let $P_n = \{\boldsymbol{u}_0, \dots, \boldsymbol{u}_{n-1}\}$ be a point set with n points in $[0,1)^s$. Then the random points of the digitally-shifted point set associated with P_n have uniform distribution on $[0,1)^s$. In particular, the randomized quasi-Monte Carlo rule obtained from this point set provides an unbiased estimator: for any integrable function f over $[0,1)^s$, $\mathbb{E}\left[\frac{1}{n}\sum_{i=0}^{n-1} f(\boldsymbol{U}_i)\right] = \mathbb{E}[f(\boldsymbol{V})]$ where $\boldsymbol{V}$ has uniform distribution on $[0,1)^s$*

Let us now define the function space in which we want to derive the mean square worst-case error. The following definition of the weighted unanchored Sobolev space is from [4].

Definition 3.13 (Weighted unanchored Sobolev space)**.** The Banach space $\mathcal{H}_{s,\boldsymbol{\gamma}} = \mathcal{W}^2_{s,\boldsymbol{\gamma}}$ with the inner product

$$\langle f, g\rangle_{\mathcal{H}_{s,\boldsymbol{\gamma}}} = \sum_{\mathfrak{u}\in\{1,\dots,s\}} \gamma_{\mathfrak{u}}^{-1} \int_{[0,1]^{|\mathfrak{u}|}} \left(\int_{[0,1]^{s-|\mathfrak{u}|}} \frac{\partial^{|\mathfrak{u}|}}{\partial \boldsymbol{y}_{\mathfrak{u}}} f(\boldsymbol{y})\, \mathrm{d}\boldsymbol{y}_{-\mathfrak{u}} \times \int_{[0,1]^{s-|\mathfrak{u}|}} \frac{\partial^{|\mathfrak{u}|}}{\partial \boldsymbol{y}_{\mathfrak{u}}} g(\boldsymbol{y})\, \mathrm{d}\boldsymbol{y}_{-\mathfrak{u}} \right) \mathrm{d}\boldsymbol{y}_{\mathfrak{u}}$$

is called the weighted unanchored Sobolev space. $\mathcal{H}_{s,\boldsymbol{\gamma}}$ is a Hilbert space.

We now state the following theorem derived from [4, Lemma 6.8].

Theorem 3.14 (Digital shift mean square worst-case error in $\mathcal{H}_{s,\boldsymbol{\gamma}}$)**.** *The digital shift mean square worst-case error in $\mathcal{H}_{s,\boldsymbol{\gamma}}$ for a digital net $\mathcal{P}_{2^k}$ with 2^k points is given by*

$$\hat{e}^2(\mathcal{H}_{s,\boldsymbol{\gamma}}, \mathcal{P}_{2^k}) = \frac{1}{2^k} \sum_{i=0}^{2^k-1} \sum_{\emptyset\neq\mathfrak{u}\subseteq\{1,\dots,s\}} \gamma_{\mathfrak{u}} \prod_{j\in\mathfrak{u}} \phi(u_{i,j})$$

where

$$\phi(u) = \begin{cases} 6^{-1} - 2^{\lfloor \log_2(u)\rfloor - 1} & \text{for } 0 < u < 1, \\ 6^{-1} & \text{for } u = 0. \end{cases}$$

Remark 3.15. Note that the space defined above is unanchored contrary to the one introduced in Definition 3.4. This is due to the fact that applying a random digital shift is somehow equivalent to averaging over all possible anchors, with the uniform distribution. Then the discrepancy no longer depends on the anchor.

3.2.3 Quality measures for interlaced digital nets

As already said above, interlaced digital nets aim at exploiting the smoothness of the function to integrate to obtain faster rate of convergences compared to classical digital nets [6]. In this section, we first present a complete orthonormal system of the Hilbert space $L^2([0,1)^s)$ called the *Walsh system*. For functions with square-integrable mixed partial derivatives up to order α, i.e. for functions in the weighted Sobolev space of smoothness α, upper bounds on the Walsh coefficients are known. This motivates the introduction of a function space $\mathcal{W}_{s,\boldsymbol{\gamma},\alpha}$ called the *weighted Walsh space of smoothness* α which contains certain weighted Sobolev spaces of smoothness α. In this space, it is possible to derive upper bounds on the worst-case error for interlaced digital nets. These upper bounds can be used as figures of merit for interlaced digital nets.

The following definitions and propositions are taken from [5, Appendix A].

Definition 3.16 (Walsh functions)**.** For an integer $k \in \mathbb{N}_0$, denote its 2-adic expansion by $k = \kappa_0 + 2\kappa_1 + \cdots + \kappa_{a-1}2^{a-1}$, with $\kappa_i \in \mathbb{F}_2$. Then the k-th 2-adic Walsh function $\mathrm{wal}_k : [0,1) \to \{-1,1\}$ is defined as

$$\mathrm{wal}_k(z) = (-1)^{z_1\kappa_0+\cdots+z_a\kappa_a}$$

for $z \in [0,1)$ with is unique 2-adic expansion $z = z_1 2^{-1} + z_2 2^{-2} + \cdots$. This definition is generalized to the multi-dimensional case by using the tensor product of functions. For $s \in \mathbb{N}$, let $\boldsymbol{z} = (z_1, \ldots, z_s) \in [0,1)^s$ and $\boldsymbol{k} = (k_1, \ldots, k_s) \in \mathbb{N}_0^s$. Then the $\boldsymbol{k}$-th 2-adic Walsh function $\mathrm{wal}_{\boldsymbol{k}} : [0,1)^s \to \{-1,1\}$ is defined as

$$\mathrm{wal}_{\boldsymbol{k}}(\boldsymbol{z}) = \prod_{j=1}^{s} \mathrm{wal}_k(z_i).$$

Proposition 3.17. *The system* $\{\mathrm{wal}_{\boldsymbol{k}}, \boldsymbol{k} \in \mathbb{N}_0^s\}$ *is an orthonormal basis of the Hilbert base* $L^2([0,1)^s)$*. In particular, for* $f \in L^2([0,1)^s)$ *, the series*

$$\sum_{\boldsymbol{l} \in \mathbb{N}_0^s} \hat{f}(\boldsymbol{l}) \mathrm{wal}_{\boldsymbol{l}}(\boldsymbol{z}) \tag{4}$$

where $\hat{f}(\boldsymbol{l})$ *is the* $\boldsymbol{l}$-th Walsh coefficient *given by*

$$\hat{f}(\boldsymbol{l}) = \int_{[0,1)^s} f(\boldsymbol{u}) \mathrm{wal}_{\boldsymbol{l}}(\boldsymbol{u}) \, \mathrm{d}\boldsymbol{u}$$

converges in the L^2 *sense towards* $f(\boldsymbol{z})$ *for* $z \in [0,1)^s$*.*

We now introduce the weighted Walsh space of smoothness α using the definition of [8]. First, we need some additional notations. For an integer $k \in \mathbb{N}$, we denote its 2-adic expansion by $k = \kappa_1 2^{a_1-1} + \cdots + \kappa_\nu 2^{a_\nu - 1}$ such that $\nu \geq 1$, $1 < a_\nu < \cdots < a_1$ and $\kappa_1, \ldots, \kappa_\nu \in \{0,1\}$. For an integer $\alpha \geq 1$, we define

$$r_\alpha(k) = 2^{-\mu_\alpha(k)},$$

where

$$\mu_\alpha(k) = a_1 + \cdots + a_{\min(\alpha,\nu)}.$$

We also define $r_\alpha(\boldsymbol{k}_\mathfrak{u}) = \prod_{j\in\mathfrak{u}} r_\alpha(\boldsymbol{k}_j)$ for $\mathfrak{u} \in \{1,\ldots,s\}$ and $\boldsymbol{k}_\mathfrak{u} \in \mathbb{N}^{|\mathfrak{u}|}$.

Definition 3.18 (Weighted Walsh space of smoothness α)**.** For smoothness α and positive weights $\boldsymbol{\gamma}$, the *weighted Walsh space of smoothness* α , noted $\mathcal{W}_{s,\boldsymbol{\gamma},\alpha}$, is the function space consisting of functions $f : [0,1)^s \to \mathbb{R}$ for which the norm

$$||f||_{\mathcal{W}_{s,\boldsymbol{\gamma},\alpha}} = \max_{\mathfrak{u}\subseteq\{1,\ldots,s\}} \gamma_\mathfrak{u}^{-1} \sup_{\boldsymbol{l}_\mathfrak{u}\in\mathbb{N}^{|\mathfrak{u}|}} \frac{|\hat{f}(\boldsymbol{l}_\mathfrak{u}, \boldsymbol{0})|}{r_\alpha(\boldsymbol{l}_\mathfrak{u})} \tag{5}$$

is finite, where $(\boldsymbol{l}_\mathfrak{u}, \boldsymbol{0})$ is the vector from $\mathbb{N}_0^s$ will all the components whose indices are not in $\mathfrak{u}$ equal 0.

Remark 3.19. From [2, Theorem 3.8], we typically have that for f verifying some smoothness conditions of order α, there exists a constant C_f such that $|\hat{f}(\boldsymbol{k})| \leq C_f r_\alpha(\boldsymbol{k})$. This upper bound justifies the normalization of the Walsh coefficients $|\hat{f}(\boldsymbol{l}_\mathfrak{u}, \boldsymbol{0})|$ in Equation (5) by $r_\alpha(\boldsymbol{l}_\mathfrak{u})$.

Remark 3.20. From [5, Appendix A.3], we have that for any function $f \in \mathcal{W}_{s,\boldsymbol{\gamma},\alpha}$, the Walsh series defined in Equation (4) converges pointwise absolutely towards f. This is necessary as we want to approximate the integral of f by an average of function evaluations and that the analysis which yields the upper-bounds on the worst-case error considers the Walsh series of the function f instead of the function f itself.

The worst-case error in $\mathcal{W}_{s,\boldsymbol{\gamma},\alpha}$ for an interlaced digital net cannot be computed directly. However, it is possible to derive upper bounds on the worst-case error which are much more practical for computations. These upper bounds can then be used as quality criterion when searching for good interlaced digital nets. The following two theorems are from [8].

Theorem 3.21 (Bound on the worst-case error in $\mathcal{W}_{s,\boldsymbol{\gamma},\alpha}$ for interlaced digital nets)**.** *Let $\mathcal{P}_{2^k}$ be an interlaced digital net with 2^k points in $[0,1)^s$ with interlacing factor d and underlying digital net $\bar{\mathcal{P}}_{2^k} = \{\boldsymbol{z}_0, \ldots, \boldsymbol{z}_{2^k-1}\}$ in $[0,1)^{ds}$. Then the worst-case error $e(\mathcal{W}_{s,\boldsymbol{\gamma},\alpha}, \mathcal{P}_{2^k})$ in $\mathcal{W}_{s,\boldsymbol{\gamma},\alpha}$ is bounded by*

$$B_{\boldsymbol{\gamma},\alpha,d,(1)}(\mathcal{P}_{2^k}) = \frac{1}{2^k} \sum_{i=0}^{2^k-1} \sum_{\emptyset\neq\mathfrak{u}\subseteq\{1,\ldots,ds\}} \bar{\gamma}_\mathfrak{u}^{(1)} \prod_{j\in} \phi_{\alpha,d,(1)}(z_{i,j}) \tag{6}$$

where

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

for any $z \in [0,1)$ *in which we set* $2^{\log_2(0)} = 0$ *and*

$$\bar{\gamma}_{\mathfrak{u}}^{(1)} = \gamma_{w(\mathfrak{u})} 2^{\alpha(2d-1)|w(\mathfrak{u})|/2} \tag{7}$$

where

$$w(\mathfrak{u}) = \{1 \leq j \leq s : \mathfrak{u} \cap \{(j-1)d+1, \dots, jd\} \neq \emptyset\} \tag{8}$$

for $\mathfrak{u} \subseteq \{1, \dots, s\}$. w *is called the* weights interlacing operator.

The previous bound on the worst-case error given in Theorem 3.21 holds for $d \leq \alpha$ and $d > \alpha$. However, there exists a tighter bound in the case $d \leq \alpha$.

Theorem 3.22 (Bound on the worst-case error in $\mathcal{W}_{s,\boldsymbol{\gamma},\alpha}$ for interlaced digital nets with interlacing factor $d \leq \alpha$)**.** *Let* $\mathcal{P}_{2^k}$ *be an interlaced digital net with* 2^k *points in* $[0,1)^s$ *with interlacing factor* d *and underlying digital net* $\bar{\mathcal{P}}_{2^k} = \{\boldsymbol{z}_0, \dots, \boldsymbol{z}_{2^k-1}\}$ *in* $[0,1)^{ds}$. *Suppose further that* $d \leq \alpha$. *Then the worst-case error* $e(\mathcal{W}_{s,\boldsymbol{\gamma},\alpha}, \mathcal{P}_{2^k})$ *in* $\mathcal{W}_{s,\boldsymbol{\gamma},\alpha}$ *is bounded by*

$$B_{\boldsymbol{\gamma},d,(2)}(\mathcal{P}_{2^k}) = \frac{1}{2^k} \sum_{i=0}^{2^k-1} \sum_{\emptyset \neq \mathfrak{u} \subseteq \{1,\dots,ds\}} \bar{\gamma}_{\mathfrak{u}}^{(2)} \prod_{j \in} \phi_{d,(2)}(z_{i,j}) \tag{9}$$

where

$$\phi_{d,(2)}(z) = \frac{2^{d-1}\left(1 - 2^{(d-1)\lfloor \log_2(z) \rfloor}(2^d - 1)\right)}{2^{d-1} - 1}$$

for any $z \in [0,1)$ *in which we set* $2^{\log_2(0)} = 0$ *and*

$$\bar{\gamma}_{\mathfrak{u}}^{(2)} = \gamma_{w(\mathfrak{u})} \prod_{j \in \mathfrak{u}} b^{-((j-1) \bmod d)-1} \tag{10}$$

for $\mathfrak{u} \subseteq \{1, \dots, s\}$.

Remark 3.23. An important remark at this point is to say that the figures of merit for interlaced digital nets derived from the two previous theorems can also be interpreted as figures of merit for the underlying digital nets. This point of view allows to reuse all the tools developed for the search of good digital nets in LatNet Builder. Instead of searching for good interlaced digital nets in dimension s, it is indeed sufficient to search for good underlying digital nets in dimension $d \times s$ where d is the interlacing factor. This is the reason for which we chose to search for underlying digital nets as explained in Remark 3.1. This approach of the issue has been very successful in limiting the implementation cost of the interlacing functionalities in LatNet Builder. In particular, as explained in section 4.3.3, the coordinate-uniform evaluation algorithms designed for digital nets do also apply to interlaced digital nets.

3.3 Exploration methods

Let us state more formally the problem we want to solve.

$$\text{Find } \boldsymbol{h}^* \in \operatorname*{argmin}_{\boldsymbol{h} \in \tilde{\mathcal{S}}_{k,s}} \mathcal{D}_q(\mathcal{P}_{2^k}(\boldsymbol{h})) \tag{11}$$

where $\tilde{\mathcal{S}}_{k,s}$ is a search space for digital nets with 2^k points in $[0,1)^s$ and $\mathcal{P}_{2^k}(\boldsymbol{h})$ is the digital net associated to $\boldsymbol{h}$. $\tilde{\mathcal{S}}_{k,s}$ is a set of s-dimensional vectors of generating values. Each vector $\boldsymbol{h}$ of $\tilde{\mathcal{S}}_{k,s}$ defines a s-dimensional vector of generating matrices.

3.3.1 Exhaustive search

Recall that $\tilde{\mathcal{S}}_{k,s}$ is a finite search space. There is no known technique to find $\boldsymbol{h}^*$ apart from exploring the whole space. Such an approach is called the *exhaustive search*. It is often not feasible in practice. For instance, for 2^{15} points in dimension 10, $|\mathcal{S}_{k,s}| \approx 10^{68}$. For an irreducible polynomial Q of degree 15, the space of polynomial lattice rules has size $|\mathcal{S}_{k,s}^{poly}| \approx 10^{43}$. The space of Sobol' nets has size $|\mathcal{S}_{k,s}^{poly}| \approx 10^{15}$. None of these spaces can be exhaustively explored in practice. So in most situations, we must forget about solving exactly Equation (11). Instead we search for $\tilde{\boldsymbol{h}}^*$ such that $D_q(\mathcal{P}_{2^k}(\tilde{\boldsymbol{h}}^*))$ and $D_q(\mathcal{P}_{2^k}(\boldsymbol{h}^*))$ are close.

3.3.2 Random search

One solution consists in using a *random search* instead of an exhaustive search. It consists in choosing $\tilde{\boldsymbol{h}}^*$ as the best $\boldsymbol{h}$ among r possibilities which are uniformly and independently sampled at random over $\tilde{\mathcal{S}}_{k,s}$ for some positive integer r.

3.3.3 Component-by-component construction

As an alternative, the so-called *component-by-component (CBC) construction* may be used. This construction was proposed by Sloan in [19]. The CBC construction was used for another class of QMC rules called lattice rules in [10]. It was also applied to polynomial lattice rules [3]. This construction has also been used in [9] for Sobol' nets. For the CBC construction, we suppose that

$$\tilde{\mathcal{S}}_{k,s} = \prod_{j=1}^{s} \tilde{\mathcal{S}}_{k,s}^{(j)}$$

and that only the j-th coordinate of $\boldsymbol{h}$ is required to compute the j-th generating matrix of $\mathcal{P}_{2^k}(\boldsymbol{h})$. This is the case for all the search spaces implemented in LatNet Builder. This approach relies on the decomposition

$$[\mathcal{D}_{q,j}(\mathcal{P}_{2^k})]^q = [\mathcal{D}_{q,j-1}(\mathcal{P}_{2^k})]^q + \sum_{\mathfrak{u}\subseteq\{1,\dots,j-1\}} \left[\gamma_{\mathfrak{u}\cup\{j\}}^q \mathcal{D}_{\mathfrak{u}\cup\{j\}}(\mathcal{P}_{2^k})\right] \tag{12}$$

where $\mathcal{D}_{q,j}(\mathcal{P}_n)$ denotes the contribution to $\mathcal{D}_g(\mathcal{P}_n)$ in Equation 3 that depends only on the first j coordinates of $\mathcal{P}_n$. In this construction, $\tilde{\boldsymbol{h}}^*$ is constructed in a component-by-component manner by a greedy algorithm: given the first $j-1$ coordinates of $\tilde{\boldsymbol{h}}^*$, choose

$$\tilde{h}_j^* \in \operatorname*{argmin}_{h\in\tilde{\mathcal{S}}_{k,s}^{(j)}} \mathcal{D}_{q,j}(\mathcal{P}_{2^k}(\tilde{\boldsymbol{h}}_{j-1}^*, h))$$

where $\tilde{\boldsymbol{h}}_{j-1}^*$ denotes the vector made of the first $j-1$ components of $\tilde{\boldsymbol{h}}^*$.

To compute $\mathcal{D}_{q,j}(\mathcal{P}_{2^k}(\tilde{\boldsymbol{h}}_{j-1}^*, h))$, we do not recompute the contribution of the first $j-1$ coordinates $\mathcal{D}_{q,j-1}(\mathcal{P}_{2^k}(\tilde{\boldsymbol{h}}_{j-1}^*))$ but instead store the result. This is useful to spare some computations.

The component-by-component construction reduces drastically the cost of exploring the search space: only a small subset is explored. Moreoever, for polynomial lattice rules and

interlaced polynomial lattice rules, it has been shown [3], [8] that under some conditions on the weights, the discrepancy of the digital nets obtained by this CBC construction achieve the optimal rate of convergence. This is an important justification for the use of this exploration method. This exploration method has a random variant in which $\tilde{h}_j^*$ is selected among r possibilities which are uniformly and independently sampled at random over $\tilde{\mathcal{S}}_{k,s}^{(j)}$ for some positive r.

4 LatNet Builder software

LatNet Builder is a C++ software which can search for various types of QMC rules using many quality measures. It has been developed at the Stochastic Simulation and Optimization Laboratory since 2012 by David Munger under the supervision of Pr. Pierre L'Ecuyer. The original software was called Lattice Builder. The reference paper for this first version of the software is [16]. In 2017, Ayman Jemel, student from Ecole Polytechnique, developed a first extension of the software for polynomial lattice rules. This year, together with Pierre Marion, also from Ecole Polytechnique, we developed a whole new extension for digital nets and interlaced digital nets. We renamed the software LatNet Builder as it handles both lattices and digital nets. LatNet Builder is to our knowledge the most complete open-source software for constructing QMC rules.

4.1 Main contributions

Our main contribution consisted in adding functionalities for searching good digital nets and interlaced digital nets. The software supports all the construction of QMC rules presented in section 2 with several search quality measures and exploration methods (section 3). All of this is new in LatNet Builder.

We also developed efficient computation methods for the quality criteria which are used in the software. One of the algorithms we designed is presented in section 4.3. The efficiency of these computations is crucial for the software as it is a bottleneck for the search.

Aside from developing the software, we also made major improvements for its distribution. Binary distributions are available for Linux and Mac OSX platforms. The documentation has been updated and extended. Interfaces with Python and with SSJ [14], a Java library for stochastic simulation, have been added. The Python interface comes with a graphical user interface based on the Jupyter ecosystem. It replaces the previous one which was obsolete. The Python interface is wrapped in a Python package which greatly facilitates the installation of the software.

4.2 Numerical experiment

In this section, we present a numerical experiment which investigates how the point sets constructed by LatNet Builder perform for numerical integration. We use a family of test functions which was introduced by Sobol' in [21]. For a given dimension s and a real number $c \in (0, 1]$,

define for $\boldsymbol{x} \in [0,1)^s$

$$f_c(\boldsymbol{x}) = \prod_{j=1}^{s} \left(1 + c \times (x_j - \frac{1}{2})\right)$$

For these functions, the ANOVA decomposition is known and equals

$$f_c(\boldsymbol{x}) = 1 + \sum_{l=1}^{s} c^l \sum_{1 \leq i_1 < \cdots < \leq s} (x_{i_1} - \frac{1}{2}) \cdots (x_{i_l} - \frac{1}{2}).$$

The variance explained by a projection $\mathfrak{u}$ equals

$$\sigma^2_{\mathfrak{u}}(f_c) = \left(\frac{c}{12}\right)^{|\mathfrak{u}|}. \tag{13}$$

For a fixed c, the fraction of variance explained by the projections of order lower than cs, i.e.

$$\frac{\sum_{\mathfrak{u} \subseteq \{1,\ldots,s\}, \atop |\mathfrak{u}| \leq cs} \sigma^2_{\mathfrak{u}}(f_c)}{\mathrm{Var}(f_c)}$$

converges towards 1 when s goes to infinity. We say that this function has an *effective dimension at the limit in the superposition sense* equal to cs. This means that the most important projections to compute the integral of f_c over $[0,1)^s$ are those of order lower than sc.

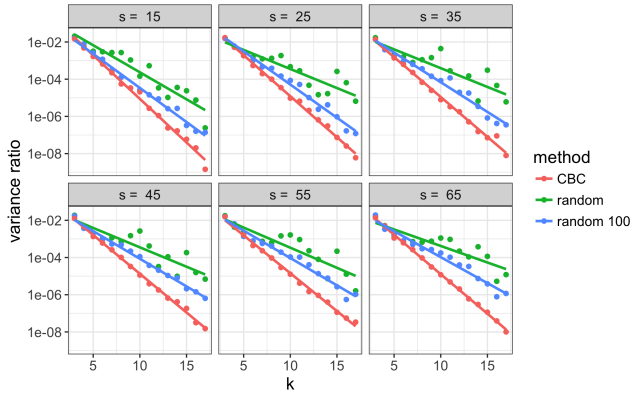
We focus on polynomial lattice rules, using the digital shift mean square worst-case error from Theorem 3.14 as a quality criteria with order dependent weights $\gamma_{|\mathfrak{u}|} = c^{|\mathfrak{u}|}$ which mimic the variance decomposition in Equation 13.

For various dimensions s between 15 and 65, various effective dimensions in the superposition sense sc between 1 and 15 and various number of points 2^k between 2^3 and 2^{17}, we estimate the variance of the classical MC estimator. We also use LatNet Builder to construct various polynomial lattice rules:

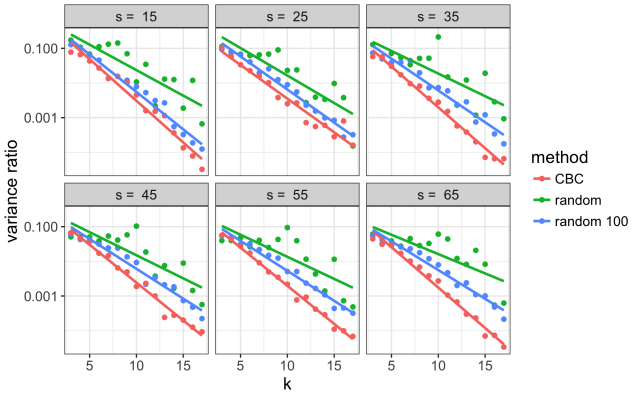
- a random polynomial lattice rule, with no optimization at all (random);
- a polynomial lattice rule using a random search with 100 samples (random 100);
- a polynomial lattice rule constructed using the CBC algorithm (CBC).

For each of this point set, we estimate the variance of the RQMC estimator using 100 independent randomizations (the randomization is a digital shift plus a linear scrambling [5]). Finally, we compute the variance ration between the RQMC estimator and the MC estimator. This ratio indicates the variance reduction provided by the QMC rules. The smaller the ratio, the greater the variance reduction. All the simulations are done using SSJ [14], a Java library for stochastic simulation developped by the Stochastic Simulation and Optimization Laboratory.

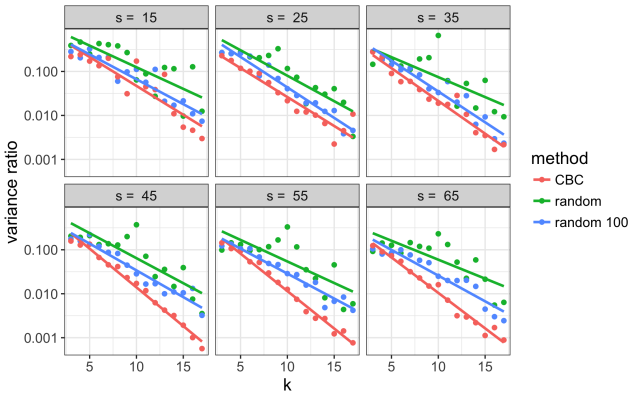
Figure 4 presents the variance ratio as a function of k for various dimensions s and effective dimensions sc. First, note that for any combinations of these parameters, the variance ratio is smaller than 1. This indicates that the RQMC estimator indeed provides a variance reduction for each of these test functions. We see that for all of these construction, there seems to be



(a) $sc = 1$

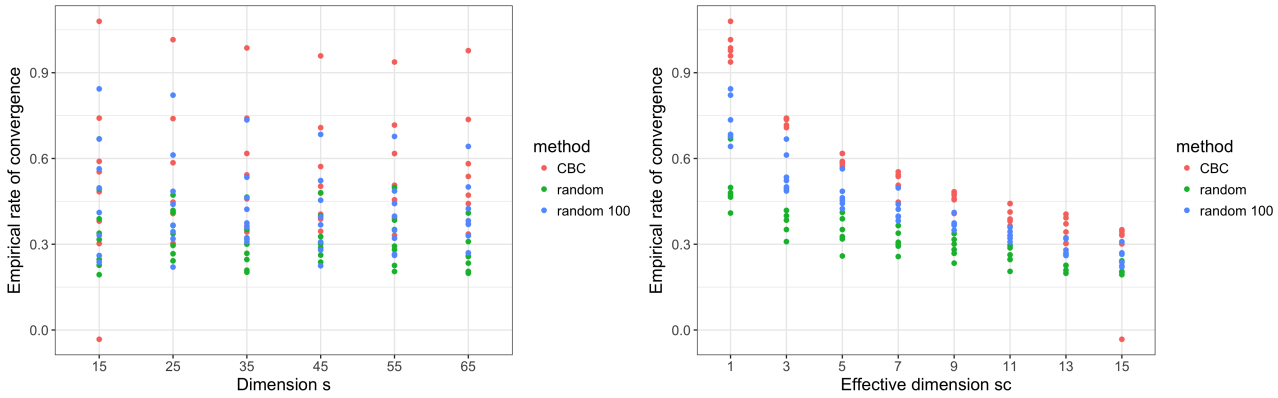


(b) $sc = 7$



(c) $sc = 13$

Figure 4: Variance ratio as a function of k for various dimensions s and effective dimensions sc



(a) as a function of s

(b) as a function of sc

Figure 5: Empirical rate of convergence for the random, random 100 and CBC construction methods

a convergence of the variance ratio towards 0. This means that the variance of the RQMC estimator converges towards 0 faster than the variance of the MC estimator. The empirical rate of convergence is better for the RQMC method.

We also added linear regression lines which show that the variance ratio converges towards zero at a $\mathcal{O}(2^{-\eta k})$ rate where η depends on the construction (random, random with 100 samples and CBC). As we might have expected, for any dimension s and effective dimension sc, the CBC construction provides the greater gain for the convergence rate. Then comes the random 100 construction and finally the random construction. This result provides a justification for the search of good QMC rules: this can increase the rate of convergence and the variance reduction.

Additionally, we present in Figure 5 the empirical rate of convergence η for various dimensions s and effective dimensions sc. First, this figure confirms what has been said before concerning the construction methods: the CBC provides a greater gain than the two others constructions. Then, we also see that the convergence rate η depends on the effective dimension sc and no so much on the dimension s. This confirms the importance of the effective dimension for QMC methods to perform well: the higher the effective dimension, the lesser the gain.

All in all, this experiment shows the interest of constructing good QMC rules for numerical integration with LatNet Builder: the integration error can be reduced compared to the classical MC integration.

4.3 Coordinate-uniform evaluation algorithms

In this section, we describe the algorithms used to compute the figures of merit presented in section 3.2. These algorithms belong to the coordinate-uniform evaluation family. The concept of coordinate-uniform algorithms comes from [16]. During the internship, we generalized these algorithms to digital nets and interlaced digital nets. The bound on the weighted star discrepancy for digital nets (Theorem 3.10) and the digital shift mean square worst-case error in the weighted unanchored Sobolev space (Theorem 3.14) are *coordinate-uniform figures of*

merit. For these figures, the projection-dependent figure of merit has a specific form which can be used to reduce the evaluation cost of the figure.

Definition 4.1 (Coordinate-uniform figure of merit)**.** A weighted figure of merit $\mathcal{D}_q$ is a *coordinate-uniform figure of merit* if there exists a function $\omega : [0,1) \to \mathbb{R}$ such that for any point set $\mathcal{P}_n$ in dimension s and any projection $\emptyset \neq \mathfrak{u} \subseteq \{1,\dots,s\}$, the projection-dependent figure of merit can be written

$$[\mathcal{D}_{\mathfrak{u}}(\mathcal{P}_n)]^q = \frac{1}{n}\sum_{i=0}^{n-1}\prod_{j\in\mathfrak{u}}\omega(u_{i,j}). \tag{14}$$

The function ω is called the *kernel* of the coordinate-uniform figure of merit $\mathcal{D}_q$.

More precisely, we want to compute such coordinate-uniform figures of merit for specific point sets called *permutation point set*.

Definition 4.2 (Permutation point set)**.** A point set $\mathcal{P}_n$ with n points in $[0,1)^s$ is a *permutation point set* if for each coordinate $1 \le j \le s$, there exists a permutation π_j of $\{0,\dots,n-1\}$ such that for $0 \le i < n$, $u_{i,j} = \pi_j(i)/n$. π_j is called the *permutation associated with the j-th coordinate of $\mathcal{P}_n$*.

A naive approach to compute a coordinate-uniform figure of merit $\mathcal{D}_{q,s}(\mathcal{P}_n)$ would be to compute independently each 2^s-1 terms in Equation (3). In terms of time and space complexities, computing the right-hand side of Equation (12) in coordinate-uniform form requires $\mathcal{O}(n)$ space and $\mathcal{O}(2^j jn)$ operations.

Alternatively, it is possible to reduce the number of operations at the expense of using more memory. The algorithms which exploit this possibility are called *coordinate-uniform evaluation algorithms* and were already investigated for QMC rules called ordinary lattice rules [16].

In this section, we first extend the framework of coordinate-uniform evaluation algorithms presented in [16] to the general case of permutation point sets. Then, we show how they can be efficiently applied to digital nets. Finally, we explain how this framework can also be used for interlaced digital nets.

Coordinate-uniform evaluation algorithms apply to any permutation point set $\mathcal{P}_n$ and coordinate-uniform figure of merit. Let $\boldsymbol{\omega} = (\omega_0,\dots,\omega_{n-1})$ be the vector with components $\omega_i = \omega(i/n)$ for $i = 0,\dots,n-1$. For $j = 1,\dots,s$, let also $\boldsymbol{\omega}^{(j)} = (\omega_{\pi_j(0)},\dots,\omega_{\pi_j(n-1)})$ resulting from the permutation π_j associated with the j-th coordinate of $\mathcal{P}_n$ applied to the components of $\boldsymbol{\omega}$.

For coordinate-uniform figures of merit, the last term of Equation (12) can be written in vector form as

$$\frac{1}{n}\sum_{i=0}^{n-1}\omega_{\pi_j(i)}\sum_{\emptyset\neq\mathfrak{u}\subseteq\{1,\dots,j-1\}}\gamma_{\mathfrak{u}\cup\{j\}}\prod_{j\in\mathfrak{u}}\omega_{\pi_j(i)} = \frac{\boldsymbol{\omega}^{(j)}}{n}\bullet\left(\sum_{\emptyset\neq\mathfrak{u}\subseteq\{1,\dots,j-1\}}\gamma_{\mathfrak{u}\cup\{j\}}\boldsymbol{q}_{\mathfrak{u}}\right)$$

where $\bullet$ denotes the scalar product, and

$$\boldsymbol{q}_{\mathfrak{u}} = \left(\prod_{k\in\mathfrak{u}}\omega_{\pi_k(0)},\dots,\prod_{k\in\mathfrak{u}}\omega_{\pi_k(n-1)}\right) = \bigodot_{k\in\mathfrak{u}}\boldsymbol{\omega}^{(k)}$$

where $\odot$ denotes the element-wise product of vectors.

All in all, we obtain for $j = 1, \dots, s$

$$[\mathcal{D}_{q,j}(\mathcal{P}_n)]^q = [\mathcal{D}_{q,j-1}(\mathcal{P}_n)]^q + \frac{\boldsymbol{\omega}^{(j)} \bullet \bar{\boldsymbol{q}}_j}{n} \tag{15}$$

$$\bar{\boldsymbol{q}}_j = \sum_{\mathfrak{u} \subseteq \{1,\dots,j-1\}} \gamma^q_{\mathfrak{u}\cup\{j\}} \boldsymbol{q}_{\mathfrak{u}} \tag{16}$$

$$\boldsymbol{q}_{\mathfrak{u}\cup\{j\}} = \boldsymbol{\omega}^{(j)} \odot \boldsymbol{q}_{\mathfrak{u}} \quad (\mathfrak{u} \subseteq \{1, \dots, j-1\}) \tag{17}$$

with initial states $\mathcal{D}_{q,0}(\mathcal{P}_n) = 0$ and $\boldsymbol{q}_\emptyset = \mathbf{1}$. $\bar{\boldsymbol{q}}_j$ is called the *weighted state*: it corresponds to a weighted sum of the *states* $\boldsymbol{q}_{\mathfrak{u}}$.

Computing $\bar{\boldsymbol{q}}_j$ given all the $\boldsymbol{q}_{\mathfrak{u}}$ for $\mathfrak{u} \subseteq \{1, \dots, j-1\}$ requires $\mathcal{O}(2^j n)$ operations and storage for all states $\boldsymbol{q}_{\mathfrak{u}}$ for $\mathfrak{u} \subseteq \{1, \dots, j-1\}$. The number of operations is indeed reduced compared to the naive approach but the memory consumption is much worse as it grows exponentially with the dimension j.

4.3.1 Specializations for some types of weights

For specific types of weights, Equation (16) can be simplified by introducing well chosen linear combinations of states as intermediate variables. These intermediate variables follow recurrence equations which avoid the individual computation of each state $\boldsymbol{q}_{\mathfrak{u}}$. This simplification thus yields a significant gain both in time and space complexities.

Product weights For product weights, we have for $\mathfrak{u} \subseteq \{1, \dots, s\}$

$$\gamma_{\mathfrak{u}} = \prod_{j \in \mathfrak{u}} \gamma_j$$

where $(\gamma_j)_{1 \le j \le s}$ are positive real numbers. Equation (16) can be rewritten

$$\bar{\boldsymbol{q}}_j = \gamma^q_j \boldsymbol{q}_{j-1}$$

where the states

$$\boldsymbol{q}_j = \sum_{\mathfrak{u} \subseteq \{1,\dots,j\}} \gamma^q_{\mathfrak{u}} \boldsymbol{q}_{\mathfrak{u}}$$

follow for $j = 1, \dots, s$ the recurrence equation

$$\boldsymbol{q}_j = (1 + \gamma^q_j \boldsymbol{\omega}^{(j)}) \boldsymbol{q}_{j-1}. \tag{18}$$

In this settings, to compute the weighted state $\bar{\boldsymbol{q}}_j$, the coordinate-evaluation algorithm requires $\mathcal{O}(n)$ operations and storage for the state $\boldsymbol{q}_{j-1}$. A single state in sufficient as the recurrence equation (18) can be computed in place.

Order-dependent weights For order-dependent weights, we have for $\mathfrak{u} \subseteq \{1, \dots, s\}$

$$\gamma_{\mathfrak{u}} = \Gamma_{|\mathfrak{u}|}$$

where $(\Gamma_l)_{0 \leq l \leq s}$ are positive real numbers. Equation (16) can be rewritten

$$\bar{\boldsymbol{q}}_j = \sum_{l=0}^{j-1} \Gamma_{l+1}^q \boldsymbol{q}_{j-1,l}$$

where the states

$$\boldsymbol{q}_{j,l} = \sum_{\substack{\mathfrak{u} \subseteq \{1,\dots,j\} \\ |\mathfrak{u}|=l}} \boldsymbol{q}_{\mathfrak{u}}$$

follow for $j = 1, \dots, s$ and $l = 1, \dots, j$ the recurrence equation

$$\boldsymbol{q}_{j,l} = \boldsymbol{q}_{j-1,l} + \boldsymbol{\omega}^{(j)} \odot \boldsymbol{q}_{j-1,l-1} \tag{19}$$

with $\boldsymbol{q}_{j,0} = \mathbf{1}$ for all j and $\boldsymbol{q}_{j,l} = \mathbf{0}$ when $l > j$. The states $\boldsymbol{q}_{j,l}$ can be computed in place: it suffices to use the recurrence equation (19) by decreasing order l. Hence, to compute the weighted state $\bar{\boldsymbol{q}}_j$ the coordinate-uniform evaluation algorithm requires $\mathcal{O}(jn)$ operations and storage for the states $\boldsymbol{q}_{j,l}$ for $l = 1, \dots, j-1$.

Product and Order-dependent weights For product order-dependent weights, we have for $\mathfrak{u} \subseteq \{1, \dots, s\}$

$$\gamma_{\mathfrak{u}} = \Gamma_{|\mathfrak{u}|} \prod_{j \in \mathfrak{u}} \gamma_j$$

where $(\Gamma_l)_{0 \leq l \leq s}$ and $(\gamma_j)_{1 \leq j \leq s}$ are positive real numbers. Equation (16) can be rewritten

$$\bar{\boldsymbol{q}}_j = \gamma_j \sum_{l=0}^{j-1} \Gamma_{l+1}^q \boldsymbol{q}_{j-1,l}$$

where the states

$$\boldsymbol{q}_{j,l} = \sum_{\substack{\mathfrak{u} \subseteq \{1,\dots,j\} \\ |\mathfrak{u}|=l}} \left(\prod_{r \in \mathfrak{u}} \gamma_r^q \right) \boldsymbol{q}_{\mathfrak{u}}$$

follow for $j = 1, \dots, s$ and $l = 1, \dots, j$ the recurrence equation

$$\boldsymbol{q}_{j,l} = \boldsymbol{q}_{j-1,l} + \gamma_j^q \boldsymbol{\omega}^{(j)} \odot \boldsymbol{q}_{j-1,l-1}$$

with $\boldsymbol{q}_{j,0} = \mathbf{1}$ for all j and $\boldsymbol{q}_{j,l} = \mathbf{0}$ when $l > j$. In this setting, the time and space complexity are the same as in the order-dependent weights setting.

4.3.2 Coordinate-uniform evaluation algorithms for digital nets

As previously explained, coordinate-uniform evaluation algorithms had already been used in computer searches for good QMC rules. However, to our knowledge, they had only been applied to ordinary lattice rules. But these algorithms do actually apply to a broader class of point sets called permutation point sets. It turns out that under some conditions, digital nets are permutation point sets. Better, the computation of the associated permutations can be done efficiently using a Gray code.

The following proposition provides a sufficient and necessary conditions for a digital net to be a permutation point set. It proofs immediately follows from Definition 2.1.

Proposition 4.3. *A digital net $\mathcal{P}_{2^k}$ with generating matrices $M_1, \dots, M_s$ of size $r \times k$ is a permutation point set if and only if $r = k$ and its generating matrices are full-rank over $\mathbb{F}_2$.*

As stated above, the computation of the associated permutations can be done efficiently using a Gray code. The idea of using Gray code to compute the points a digital net was first used in [1] for the special case of Sobol' nets. Let us first recall the definition of Gray codes.

Definition 4.4 (Binary Gray code)**.** A binary Gray code with k bits is a permutation G_k of $\{0, \dots, 2^k - 1\}$ such that two successive values only differ by one binary digit. The position of the binary digit which differs between $G_k(i-1)$ and $G_k(i)$ is noted $\delta_{G_k}(i)$. In other words, $\delta_{G_k}(i)$ is the unique integer between 1 and k such that $G_k(i) = G_k(i-1) \oplus 2^{\delta_{G_k}(i)-1}$ where $\oplus$ denotes the bitwise exclusive-or operator.

Let us now state our algorithm to compute the permutations associated with the coordinates of a digital net.

Proposition 4.5. *Let $\mathcal{P}_{2^k}$ be a digital net with full-rank generating matrices $M_1, \dots, M_s$ of size $k \times k$. For $1 \le j \le s$ and $1 \le r \le k$, denote by $M_j(r)$ the integer whose binary representation is the r-th column of M_j. Then, for a Gray code with k bits G_k, define for each $1 \le j \le s$, $\tilde{\pi}_j(0) = 0$ and for $i = 1, \dots, 2^k - 1$*

$$\tilde{\pi}_j(i) = M_j(\delta_{G_k}(i)) \oplus \tilde{\pi}_j(i-1). \tag{20}$$

Then, $\mathcal{P}_{2^k}$ is a permutation point set and for $j = 1, \dots, s$, the permutation associated to its j-th coordinate $\pi_j = \tilde{\pi}_j \circ G_k^{-1}$.

Proof. It is sufficient to note that the $\tilde{\pi}_j$ defined by Equation (20) verify

$$\tilde{\boldsymbol{\pi}}_j(i) = M_l \times \boldsymbol{G}_k(i)$$

where $\tilde{\boldsymbol{\pi}}_j(i)$ and $\boldsymbol{G}_k(i)$ denote the k-dimensional vectors over $\mathbb{F}_2$ corresponding to the binary digits of $\tilde{\pi}_j(i)$ and $G_k(i)$ respectively. Than the result follows from Definition 2.1 and Proposition 4.3. □

Remark 4.6. As Equation (15) does not depend on the order of the points, we can rearrange the components of the permuted kernel $\boldsymbol{\omega}^{(j)}$ and of the states $\boldsymbol{q}_u$ by applying the permutation G_k without changing the results of the computation. This allows to skip the right-composition by G_k^{-1} in the definition of π_j.

Using Proposition 4.5, it is therefore possible to compute the permuted kernel $\boldsymbol{\omega}^{(j)}$ associated to a generating matrix M_j using $\mathcal{O}(2^k)$ operations instead of $\mathcal{O}(k2^k)$ operations when using the naive approach obtained by following Definition 2.1.

4.3.3 Coordinate-uniform evaluation algorithms for interlaced digital nets

As explained in Remark 3.23, the figures of merit for interlaced digital nets we have presented in section 3.2.3 can be interpreted as figures of merit for digital nets in higher dimension. Hence, in the remaining of this section, we consider $B_{\boldsymbol{\gamma},\alpha,d,(1)}$ and $B_{\boldsymbol{\gamma},d,(2)}$ as figures of merit for digital nets in dimension $d \times s$. We have also taken care of formulating Equations (6) and (9) to make clear that these figures of merit are coordinate-uniform. Hence, the coordinate-uniform evaluation algorithms do apply to their computation.

However, the transformation of the weights $\boldsymbol{\gamma}$ in dimension s into the weights $\bar{\boldsymbol{\gamma}}$ in dimension $d \times s$ through the weights interlacing operator w (Equations (7) and (10)) does not preserve the type of weights. Hence, the efficient specializations for product, order-dependent and product order-dependent weights presented in section 4.3.1 cannot be used for these figures of merit. However, some kind of structure can still be observed for these interlaced weights. We call the weights resulting from the interlacing of product order-dependent weights *interlaced product order-dependent weights*. They have the general form

$$\bar{\gamma}_{\mathfrak{u}} = \Gamma_{|w(\mathfrak{u})|} \times \tilde{\Gamma}_{|w(\mathfrak{u})|} \times \prod_{j \in w(\mathfrak{u})} \gamma_j \times \prod_{j \in \mathfrak{u}} \tilde{\gamma}_j$$

for $\mathfrak{u} \subseteq \{1, \dots, ds\}$, where $(\Gamma_l)_{0 \le l \le s}$ and $(\gamma_j)_{1 \le j \le s}$ are the order-dependent component and the product component of $\boldsymbol{\gamma}$ respectively and

$$\tilde{\Gamma}_l = 2^{\alpha(2d-1)l/2} \quad \tilde{\gamma}_j = 1$$

for $B_{\boldsymbol{\gamma},\alpha,d,(1)}$ and

$$\tilde{\Gamma}_l = 1 \quad \tilde{\gamma}_j = b^{-((j-1) \bmod d)-1}$$

for $B_{\boldsymbol{\gamma},d,(2)}$ with $0 \le l \le s$ and $1 \le j \le ds$.

For interlaced product order-dependent weights, we can derive formulas equivalent to those already known for more classical types of weights. To our knowledge, the use of coordinate-uniform evaluation algorithms for these figures of merit with product order-dependent weights is new.

Equation (16) can be rewritten for $j = 0, \dots, d-1$ and $k = 0, \dots, d-1$

$$\bar{\boldsymbol{q}}_{jd+k+1} = \gamma_{j+1}^q \left(\sum_{i=0}^{k} \boldsymbol{e}_{k,i} \left(\tilde{\gamma}_{jd+1}^q \boldsymbol{\omega}^{(jd+1)}, \dots, \tilde{\gamma}_{jd+k}^q \boldsymbol{\omega}^{(jd+k)} \right) \right) \odot \sum_{l=0}^{j} \Gamma_{l+1}^q \tilde{\Gamma}_{l+1}^q \boldsymbol{q}_{j,l} \tag{21}$$

where the states

$$\boldsymbol{q}_{j,l} = \sum_{\substack{\mathfrak{u} \subseteq \{1,\dots,dj\} \\ |w(\mathfrak{u})|=l}} \left(\prod_{r \in w(\mathfrak{u})} \gamma_r^q \right) \boldsymbol{q}_{\mathfrak{u}}$$

follow for $j = 1, \dots, s$ and $l = 1, \dots, j$ the recurrence equation

$$\boldsymbol{q}_{j,l} = \boldsymbol{q}_{j-1,l} + \gamma_j^q \left(\sum_{i=1}^{d} \boldsymbol{e}_{d,i} \left(\tilde{\gamma}^q_{(j-1)d+1} \boldsymbol{\omega}^{((j-1)d+1)}, \dots, \tilde{\gamma}^q_{jd} \boldsymbol{\omega}^{(jd)} \right) \right) \odot \boldsymbol{q}_{j-1,l-1} \tag{22}$$

with $\boldsymbol{q}_{j,0} = \mathbf{1}$ for all j and $\boldsymbol{q}_{j,l} = \mathbf{0}$ when $l > j$ and $\boldsymbol{e}_{k,i}$ denotes the i -th elementary symmetric polynomial in k variables applied element-wise to vectors using the $\odot$ product.

Additionally, to efficiently compute Equations (21) and (22), we also have the relation

$$\sum_{i=0}^{k+1} \boldsymbol{e}_{k+1,i}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_{k+1}) = (1 + \boldsymbol{x}_{k+1}) \odot \sum_{i=0}^{k} \boldsymbol{e}_{k,i}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_k). \tag{23}$$

As for classical product order-dependent weights, the states can be computed in place by using the recurrence equation (22) by decreasing order l. The only additional cost is to store and update $\sum_{i=0}^{k} \boldsymbol{e}_{k,i} \left(\tilde{\gamma}^q_{jd+1} \boldsymbol{\omega}^{(jd+1)}, \dots, \tilde{\gamma}^q_{jd+k} \boldsymbol{\omega}^{(jd+k)} \right)$ using Equation (23). This does not change the overall space and time complexities: to compute the weighted state $\bar{\boldsymbol{q}}_{jd+k+1}$, we only need $\mathcal{O}((jd + k + 1)n)$ operations and storage.

This generalization of coordinate-uniform evaluation algorithms to interlaced digital nets enables to reuse all the machinery of LatNet Builder for interlaced digital nets, making them easy to search for.

5 Conclusion

During this internship, we developed an extension of LatNet Builder for constructing good digital nets and interlaced digital nets for numerical integration. Thanks to this new feature, LatNet Builder is now the most complete open-source software to construct QMC rules. The new algorithms we designed make the software efficient and reaching state-of-the-art. The software is available online with a complete documentation and awaits for users. It is easy to install and to use.

In the near future, the laboratory will try to publicize the software. QMC and RQMC methods have proven efficient for several applications[12, 15, 13] but remain quite confidential outside of the community. LatNet Builder could contribute to the visibility of these techniques.

Aside from this question, much work remains to be done to further investigate the construction of good QMC rules. In particular, the choice of the quality criterion and of its various parameters such as the weights is far from being an easy task for non-expert users. Providing techniques and guidelines would greatly reduce the complexity of LatNet Builder for its potential future users.

References

[1] I. A. Antonov and V. M. Saleev. An economic method of computing LP_τ-sequences. *Zh. Vychisl. Mat. i. Mat. Fiz.*, 19:243–245, 1979. In Russian.

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

[3] J. Dick, F. Y. Kuo, F. Pillichshammer, and I. H. Sloan. Construction algorithms for polynomial lattice rules for multivariate integration. *Mathematics of Computation*, 74(252):1895–1921, 2005.

[4] J. Dick, F. Y. Kuo, and I. H. Sloan. High dimensional integration—the quasi-Monte Carlo way. *Acta Numerica*, 22:133–288, 2013.

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

[6] Josef Dick. 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.

[7] Michael Gnewuch, Anand Srivastav, and Carola Winzen. Finding optimal volume subintervals with k points and calculating the star discrepancy are np-hard problems. *Journal of Complexity*, 25(2):115 – 127, 2009.

[8] Takashi Goda. Good interlaced polynomial lattice rules for numerical integration in weighted Walsh spaces. 285, 06 2013.

[9] S. Joe and F. Y. Kuo. Constructing Sobol sequences with better two-dimensional projections. *SIAM Journal on Scientific Computing*, 30(5):2635–2654, 2008.

[10] F. Y. Kuo and S. Joe. Component-by-component construction of good lattice rules with a composite number of points. *Journal of Complexity*, 18(4):943–976, 2002.

[11] F. Y. Kuo, C. Schwab, and I. H. Sloan. Quasi-Monte Carlo methods for high dimensional integration: the standard (weighted Hilbert space) setting and beyond. *The ANZIAM Journal*, 53:1–37, 2011.

[12] P. L'Ecuyer. Quasi-Monte Carlo methods with applications in finance. *Finance and Stochastics*, 13(3):307–349, 2009.

[13] P. L'Ecuyer. Randomized quasi-Monte Carlo: An introduction for practitioners. In P. W. Glynn and A. B. Owen, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2016*, Berlin, 2017. Springer-Verlag. to appear.

[14] P. L'Ecuyer, L. Meliani, and J. Vaucher. SSJ: A framework for stochastic simulation in Java. In E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, editors, *Proceedings of the 2002 Winter Simulation Conference*, pages 234–242. IEEE Press, 2002.

[15] P. L'Ecuyer and D. Munger. On figures of merit for randomly-shifted lattice rules. In H. Woźniakowski and L. Plaskota, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2010*, pages 133–159, Berlin, 2012. Springer-Verlag.

[16] P. L'Ecuyer and D. Munger. Algorithm 958: Lattice builder: A general software tool for constructing rank-1 lattice rules. *ACM Trans. on Mathematical Software*, 42(2):Article 15, 2016.

[17] Kent Morrison et al. Integer sequences and matrices over finite fields. *J. Integer Seq*, 9(2):06–2, 2006.

[18] H. Niederreiter. Low-discrepancy point sets obtained by digital constructions over finite fields. *Czechoslovak Math. Journal*, 42:143–166, 1992.

[19] I. H. Sloan, F. Y. Kuo, and S. Joe. On the step-by-step construction of quasi-Monte Carlo integration rules that achieve strong tractability error bounds in weighted Sobolev spaces. *Mathematics of Computation*, 71(240):1609–1640, 2002.

[20] I. M. Sobol'. 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.

[21] I. M. Sobol and D. I. Asotsky. One more experiment on estimating high-dimensional integrals by quasi-monte carlo methods. *Math. Comput. Simul.*, 62(3-6):255–263, March 2003.