Metadata of the chapter that will be visualized in SpringerLink

Book Title	Monte Carlo and Quasi-Monte Carlo 2024	
Series Title		
Chapter Title	Quasi-Monte Carlo Methods: What, Why, and How?	
Copyright Year	2026	
Copyright HolderName	The Author(s), under exclusive license to Springer Nature Switzerland AG	
Corresponding Author	Family Name	**Hickernell**
	Particle	
	Given Name	**Fred J.**
	Prefix	
	Suffix	
	Role	
	Division	Department of Applied Mathematics
	Organization	Illinois Institute of Technology
	Address	Chicago, IL, USA
	Email	hickernell@illinoistech.edu
Author	Family Name	**Kirk**
	Particle	
	Given Name	**Nathan**
	Prefix	
	Suffix	
	Role	
	Division	Department of Applied Mathematics
	Organization	Illinois Institute of Technology
	Address	Chicago, IL, USA
	Email	nkirk@illinoistech.edu
Author	Family Name	**Sorokin**
	Particle	
	Given Name	**Aleksei G.**
	Prefix	
	Suffix	
	Role	
	Division	Department of Applied Mathematics
	Organization	Illinois Institute of Technology
	Address	Chicago, IL, USA
	Email	asorokin@hawk.illinoistech.edu

Abstract

Many questions in quantitative finance, uncertainty quantification, and other disciplines are answered by computing the population mean, $\mu := \mathbb{E}(Y)$, where instances of $Y := f(\boldsymbol{X})$ may be generated by numerical simulation and $\boldsymbol{X}$ has a simple probability distribution. The population mean can be approximated by the sample mean, $\hat{\mu}_n := n^{-1} \sum_{i=0}^{n-1} f(\boldsymbol{x}_i)$, for a well chosen sequence of nodes, $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots\}$, and a sufficiently

large sample size, n. Computing μ is equivalent to computing a d-dimensional integral, $\int f(\boldsymbol{x})\varrho(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$, where ϱ is the probability density for $\boldsymbol{X}$. Quasi-Monte Carlo methods replace independent and identically distributed sequences of random vector nodes, $\{\boldsymbol{x}_i\}_{i=0}^{\infty}$, by low-discrepancy sequences. This accelerates the convergence of $\hat{\mu}_n$ to μ as $n \to \infty$. This tutorial describes low-discrepancy sequences and their quality measures. We demonstrate the performance gains possible with quasi-Monte Carlo methods. Moreover, we describe how to formulate problems to realize the greatest performance gains using quasi-Monte Carlo. We also briefly describe the use of quasi-Monte Carlo methods for problems beyond computing the mean, μ.

Keywords (separated by '-')	Low-discrepancy - Randomization - Sample mean - Simulation

Quasi-Monte Carlo Methods: What, Why, and How?

Fred J. Hickernell, Nathan Kirk, and Aleksei G. Sorokin

Abstract Many questions in quantitative finance, uncertainty quantification, and other disciplines are answered by computing the population mean, $\mu := \mathbb{E}(Y)$, where instances of $Y := f(\boldsymbol{X})$ may be generated by numerical simulation and $\boldsymbol{X}$ has a simple probability distribution. The population mean can be approximated by the sample mean, $\hat{\mu}_n := n^{-1} \sum_{i=0}^{n-1} f(\boldsymbol{x}_i)$, for a well chosen sequence of nodes, $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots\}$, and a sufficiently large sample size, n. Computing μ is equivalent to computing a d-dimensional integral, $\int f(\boldsymbol{x})\varrho(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$, where ϱ is the probability density for $\boldsymbol{X}$. Quasi-Monte Carlo methods replace independent and identically distributed sequences of random vector nodes, $\{\boldsymbol{x}_i\}_{i=0}^{\infty}$, by low-discrepancy sequences. This accelerates the convergence of $\hat{\mu}_n$ to μ as $n \to \infty$. This tutorial describes low-discrepancy sequences and their quality measures. We demonstrate the performance gains possible with quasi-Monte Carlo methods. Moreover, we describe how to formulate problems to realize the greatest performance gains using quasi-Monte Carlo. We also briefly describe the use of quasi-Monte Carlo methods for problems beyond computing the mean, μ.

AQ1

Keywords Low-discrepancy · Randomization · Sample mean · Simulation

F. J. Hickernell (✉) · N. Kirk · A. G. Sorokin
Department of Applied Mathematics, Illinois Institute of Technology, Chicago, IL, USA
e-mail: hickernell@illinoistech.edu

N. Kirk
e-mail: nkirk@illinoistech.edu

A. G. Sorokin
e-mail: asorokin@hawk.illinoistech.edu

C. Lemieux and B. Feng (eds.), *Monte Carlo and Quasi-Monte Carlo 2024*,
Springer Proceedings in Mathematics & Statistics 522,
https://doi.org/10.1007/978-3-032-10590-5_1

610417_1_En_1_Chapter-online ☑ TYPESET ☐ DISK ☐ LE ☑ CP Disp.:13/12/2025 Pages: 37 Layout: T1-Standard

1 Introduction

There are many settings where key underlying quantities that affect the outcome are unknown, e.g.,

- future market forces, which affect financial risk,
- the porosity field of rock, which affects the extraction of oil or gas, or
- elementary particle interactions in a high energy physics experiment, which affect what is observed by detectors.

In such situations, the unknown inputs are often modeled using random vector variables or stochastic processes. Computations are performed by generating a multitude of possible outcomes informed by the assumed probability distribution of the input. These are used to estimate the mean, quantiles, and/or probability distribution of the outcome. This is the essence of the Monte Carlo (MC) method.

In mathematical terms, the random outcome is $Y := f(\boldsymbol{X})$, where $\boldsymbol{X}$ is a vector random input. Given an input, $\boldsymbol{x}$, the corresponding outcome $y = f(\boldsymbol{x})$ can be computed by an algorithm, whose complexity makes f a *black box*, or at least a *gray box*. The user selects a sequence of input values, also known as data sites or *nodes*, $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots$, which lead to observed outcomes $y_0 = f(\boldsymbol{x}_0)$, $y_1 = f(\boldsymbol{x}_1), \ldots$. The y_i may be used to approximate $\mathbb{E}(Y)$, quartiles of Y, the probability density of Y, and other quantities of interest.

Simple Monte Carlo chooses the sequence of nodes, $\{\boldsymbol{x}_i\}_{i=0}^{\infty}$, to be independent and identically distributed (IID). Quasi-Monte Carlo (qMC) chooses the nodes differently, so that their empirical distribution approximates well the true probability distribution of $\boldsymbol{X}$. The difference between the true and empirical distributions is called a *discrepancy*, and the node sequences used in qMC are called low-discrepancy (LD) sequences.

This tutorial describes what qMC is, why we would want to use qMC, and how qMC can be implemented well. The next section illustrates by example the advantages of qMC. This is followed by a description of how deterministic LD sequences are constructed (Sect. 3) and randomizations that preserve their low-discrepancy (Sect. 4). We describe various measures of discrepancy (Sect. 5). We explain how to decide what sample size, n, is sufficient to meet the user's error requirements (Sect. 6). We then discuss how to rewrite the problem of interest in a qMC-friendly way (Sect. 7). We briefly discuss applications of qMC beyond computing the mean (Sect. 8), and finish with a short conclusion (Sect. 9). Inevitably, this tutorial is not comprehensive of the full breadth of over a century of development in qMC and uniform distribution theory, dating back to Hermann Weyl's seminal 1916 paper [91]. Instead, our goal is to provide a solid foundation in the core concepts along with practical guidance—especially for the newcomer—and to spark further interest and study in the field.

2 An Illustration of Quasi-Monte Carlo

We illustrate the benefits of qMC with an example from Keister [46], motivated by computational physics:

$$\mu := \int_{\mathbb{R}^d} \cos(\|\boldsymbol{t}\|_2) \exp(-\|\boldsymbol{t}\|_2^2)\, \mathrm{d}\boldsymbol{t}, \tag{1}$$

where $\|\boldsymbol{t}\|_2 := \sqrt{t_1^2 + \cdots + t_d^2}$. This integral may be evaluated numerically by re-writing it in spherical coordinates as

$$\mu = \frac{2\pi^{d/2}}{\Gamma(d/2)} \int_0^\infty \cos(r) \exp(-r^2)\, r^{d-1} \mathrm{d}r, \tag{2}$$

where $2\pi^{d/2}/\Gamma(d/2)$ is the surface area of the sphere in d dimensions, and Γ is the Gamma function. The resulting one-dimensional integral is amenable to quadrature methods. This non-trivial test case with a true value that can be easily calculated allows us to compute the numerical errors of various cubature schemes and contrast their performance.

For the purpose of this illustration, we work with μ in its original form, (1) rather than (2), and approximate it by a sample mean,

$$\hat{\mu}_n := \frac{1}{n} \sum_{i=0}^{n-1} f(\boldsymbol{x}_i). \tag{3}$$

This does require some preparation to identify a suitable f, which is discussed in generality in Sect. 7.

This μ in (1) is the expectation of $Y := g(\boldsymbol{T}) := \pi^{d/2} \cos(\|\boldsymbol{T}\|)$, where $T_1, \ldots, T_d$ are IID Gaussian (normal) random variables with zero mean and variance $1/2$, i.e., $\boldsymbol{T} := (T_1, \ldots, T_d) \sim \mathcal{N}(\boldsymbol{0}, \mathrm{I}/2)$:

$$\mu = \int_{\mathbb{R}^d} \underbrace{\pi^{d/2} \cos(\|\boldsymbol{t}\|)}_{=:g(\boldsymbol{t})} \underbrace{\frac{\exp(-\|\boldsymbol{t}\|^2)}{\pi^{d/2}}}_{\text{density of } \mathcal{N}(\boldsymbol{0}, \mathrm{I}/2)} \mathrm{d}\boldsymbol{t} = \mathbb{E}(Y) = \mathbb{E}[g(\boldsymbol{T})]. \tag{4}$$

Virtually all LD sequences underlying qMC are defined to approximate the standard uniform distribution, $\mathcal{U}[0,1]^d$. Thus, we perform a variable transformation $\boldsymbol{t} = \big(\Phi^{-1}(x_1), \ldots, \Phi^{-1}(x_d)\big)/\sqrt{2}$, where Φ is the cumulative distribution function of the standard Gaussian random variable. This reimagines the integral μ as the expectation of a function, f, of a standard uniform random variable:

$$\mu = \int_{[0,1]^d} \underbrace{\pi^{d/2} \cos\Bigl(\bigl\|\bigl(\Phi^{-1}(x_1), \ldots, \Phi^{-1}(x_d)\bigr)\bigr\|/\sqrt{2}\Bigr)}_{=: f(\boldsymbol{x})} \, \mathrm{d}\boldsymbol{x}$$

$$= \mathbb{E}[Y] = \mathbb{E}[f(\boldsymbol{X})], \qquad \boldsymbol{X} \sim \mathcal{U}[0,1]^d. \quad (5)$$

Now we can apply IID MC, qMC, and other computational methods to approximate μ by the sample mean,

$$\hat{\mu}_n = \frac{1}{n} \sum_{i=0}^{n-1} f(\boldsymbol{x}_i) = \frac{1}{n} \sum_{i=0}^{n-1} \pi^{d/2} \cos\Bigl(\bigl\|\bigl(\Phi^{-1}(x_{i1}), \ldots, \Phi^{-1}(x_{id})\bigr)\bigr\|/\sqrt{2}\Bigr). \quad (6)$$

Consider the specific case of $d = 6$ for which $\mu = -2.327303729298\ldots$ (by quadrature). We approximate μ by $\hat{\mu}_n$ for different, $\{\boldsymbol{x}_i\}_{i=0}^{n-1}$, for various sample sizes, n, and plot the relative errors, $|(\mu - \hat{\mu}_n)/\mu|$ in Fig. 1. The three kinds of nodes are

i. Cartesian grids, $\{1/(2m), \ldots, (2m-1)/(2m)\}^6$, for $m = 2, 3, \ldots$ and $n = m^6$ (blue dots),
ii. IID sequences with arbitrary n (orange diamonds), and
iii. Randomized LD (Sobol') sequences with $n = 1, 2, \ldots, 2^m, \ldots$ (green squares).

There are 50 replications for each of the latter two cases with the average errors being plotted as trend lines.

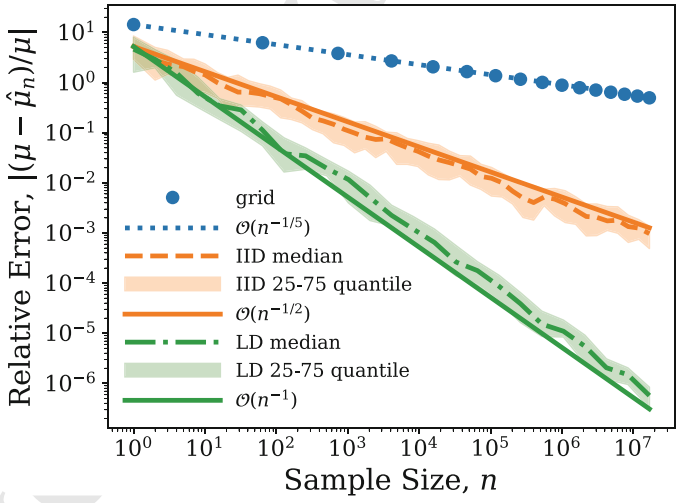


Fig. 1 The relative error of approximating μ defined in (5) by the sample mean, $\hat{\mu}_n$, defined in (6) for various choices of nodes, $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots$. Grids have the largest error and LD nodes have the smallest error

Note the following from this example:

- This integral is not particularly easy to evaluate numerically. Even the best choice of nodes requires at least $n = 100$ to get a relative error below 10%.
- *Grids.* Although they may be attractive for low-dimensional problems, e.g., $d = 1$, 2, or 3, grids do poorly for this modest dimension, $d = 6$. Figure 2 compares a 64 point grid with $d = 2$ and 6. For $d = 6$, the possible sample sizes, n, are quite sparse, as shown in Fig. 1.
  Moreover, grids are not naturally extensible. To move from $n = m^6$ nodes to $n = (m+1)^6$ nodes, one must discard the original nodes.
  Choosing the nodes to lie on a grid corresponds to a tensor product midpoint cubature rule, which would normally be expected to have an error of $O(n^{-2/d})$ for general d. The error decay of $O(n^{-1/5})$ rather than $O(n^{-2/6})$ for this example may be due to the lack of smoothness of f near the boundaries of the unit cube.
- *IID.* Simple MC is a substantial improvement over grid nodes. For IID nodes the root mean squared error is

$$\sqrt{\mathbb{E}[(\mu - \hat{\mu})^2]} = \sqrt{\mathrm{Var}(\hat{\mu})} = \sqrt{\frac{\mathrm{Var}(f(\boldsymbol{X}))}{n}} = \frac{\mathrm{Std}(f(\boldsymbol{X}))}{n^{1/2}}, \tag{7}$$

  where Var denotes the variance and Std the standard deviation. This $O(n^{-1/2})$ decay is observed in Fig. 1. For simple MC the sample size, n, can be any positive integer without affecting the rate of decay of the error.
  Whereas grid points collapse on top of one another when viewed in low-dimensional projections (Fig. 2), all IID nodes may be seen when viewed in any lower dimensional projection, as seen in Fig. 3. The disadvantage of IID nodes is that they form clusters and leave gaps. This is because the position of any one node is independent of the position of the others.
- *LD.* The error of qMC methods decays nearly like $O(n^{-1})$, which for this example correponds to a reduction in error of several orders of magnitude compared to simple MC for large enough n. Typically, LD sequences have preferred sample

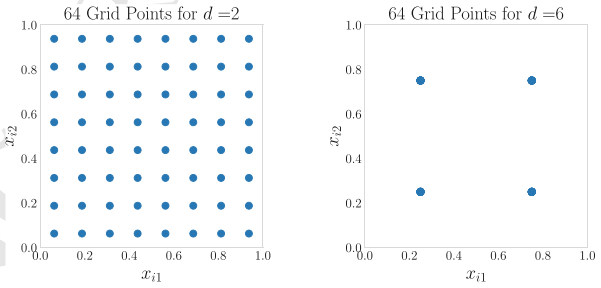


Fig. 2 Although a two-dimensional grid covers the unit square rather well, a d-dimensional grid for modest d does *not* cover the unit cube well. For example, one can only see four distinct nodes in a two dimensional projection of a six-dimensional grid with 64 nodes

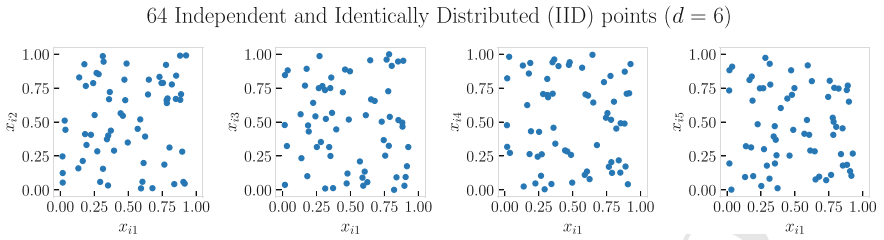


Fig. 3 IID nodes cover the unit cube better than grid nodes, although one does observe clusters and gaps. In any projection there is a similar looking distribution of all 64 nodes

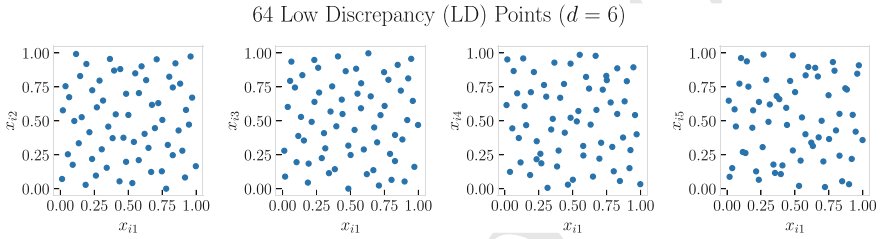


Fig. 4 LD nodes cover the unit cube even better than IID nodes or grids. In any projection there is a similar looking distribution of all 64 nodes

sizes. This particular LD sequence is a Sobol' sequence, where the preferred sample sizes are non-negative integer powers of 2.

Figure 4 shows typical two-dimensional projections of a size 64 LD node set. Visually, these nodes fill the unit cube better than IID nodes. A quantitative measure of this, the discrepancy, is defined in Sect. 5. Constructions of LD node sequences are explained in the next section.

The numerical computations used to generate the figures in this chapter may be reproduced using the Jupyter notebook in [25]. The Python library `qmcpy` [10] is used to generate the LD node sequences.

3 LD Sequence Constructions

This section introduces some of the most popular LD sequences. What sets these apart from IID sequences is that the nodes are deliberately chosen and highly correlated, whether they be deterministic or randomized. The Python library `qmcpy` [10] contains many of the constructions discussed here, as well as the stopping criteria discussed in Sect. 6.

3.1 Lattice Sequences

One of the simplest LD constructions is the family of good lattice points [16, 81]. As a finite node set, they are defined as

$$\boldsymbol{x}_i^{\text{lat}} = i\boldsymbol{h}/n \pmod{\mathbf{1}}, \qquad i = 0, \ldots, n-1, \tag{8}$$

where $\boldsymbol{h}$ is a well chosen d-dimensional integer vector. Figure 5 illustrates this construction for $n = 16$ and $\boldsymbol{h} = (1, 11)$. Note that the set $\{\boldsymbol{x}_0^{\text{lat}}, \ldots, \boldsymbol{x}_{n-1}^{\text{lat}}\}$ defined in (8) is closed under addition modulo $\mathbf{1}$ so that it forms a group.

One disadvantage of this construction is that it is not extensible. For the example in Fig. 5, the first 8 nodes do not fill the unit square well. However, the set of nodes defined by even i do a reasonable job. This suggests a method for defining extensible lattice sequences that was proposed independently in [31, 64].

An extensible lattice sequence in one dimension is defined by the van der Corput sequence in base b, $\{\phi_b(0), \phi_b(1), \ldots\}$, where ϕ_b is the radical inverse function and essentially reflects the digits of the integer i in base b about the decimal point [18, Chap. 3]. For example, $\phi_2(6) = \phi_2(110_2) = {}_20.011 = 3/8$. In general,

$$\phi_b(i_0 + i_1 b + i_2 b^2 + \cdots) := i_0 b^{-1} + i_1 b^{-2} + i_2 b^{-3} + \cdots \in [0, 1) \quad \text{where } i_0, i_1, \ldots \in \{0, \ldots b-1\}. \tag{9}$$

For all non-negative integers, m, the first $n = b^m$ nodes in the van der Corput sequence correspond to the evenly spaced nodes $\{0, b^{-m}, \ldots, 1 - b^{-m}\}$—albeit in a different order.

To construct an extensible lattice, we replace i/n in (8) by $\phi_b(i)$ to get

$$\boldsymbol{x}_i^{\text{lat}} = \phi_b(i)\boldsymbol{h} \pmod{\mathbf{1}}, \qquad i = 0, 1, \ldots. \tag{10}$$

This reordering of nodes from the original construction allows us to preserve the lattice structure for the first b^m nodes for any non-negative integer m. That is, $\{\boldsymbol{x}_0^{\text{lat}}, \ldots, \boldsymbol{x}_{b^m-1}^{\text{lat}}\}$ is a closed under addition modulo $\mathbf{1}$.

Figure 6 shows the first 4, 8, and 16 nodes of an extensible lattice in base 2 with the same generator as in Fig. 5. The blue dots are a copy of the nodes in the plot to the left, and the the orange diamonds correspond to a shifted copy of the blue dots modulo $\mathbf{1}$. For the left plot, the shift is $\phi_2(2)\boldsymbol{h} = (1, 11)/4 = (0.25, 0.75)$. For the middle plot, the shift is $\phi_2(4)\boldsymbol{h} = (1, 11)/8 = (0.125, 0.375)$. For the right plot, the shift is $\phi_2(8)\boldsymbol{h} = (1, 11)/16 = (0.0625, 0.6875)$.

For the example in Fig. 6, increasing n beyond 16 repeats the original nodes because the generating vector, $\boldsymbol{h}$, contains integers all less than 16. Obtaining a truly extensible lattice sequence requires that $\boldsymbol{h}$ be a vector of generalized integers—essentially integers with infinite numbers of nonzero digits—as explained in [36], where the existence of good generating vectors for infinite lattice sequences is also proved. In practice, one searches computationally for a $\boldsymbol{h}$ that produces good set

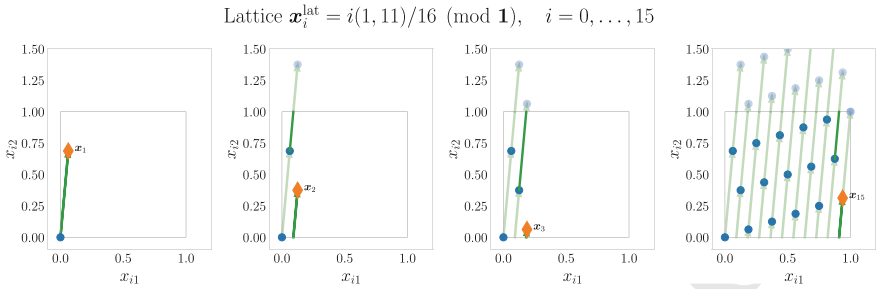


Fig. 5 The construction of a good lattice node set in two dimensions with 16 nodes is obtained by moving $\boldsymbol{h}/n$ beyond the present node and wrapping around the boundaries of the square (hypercube in d dimensions) until one returns to the origin

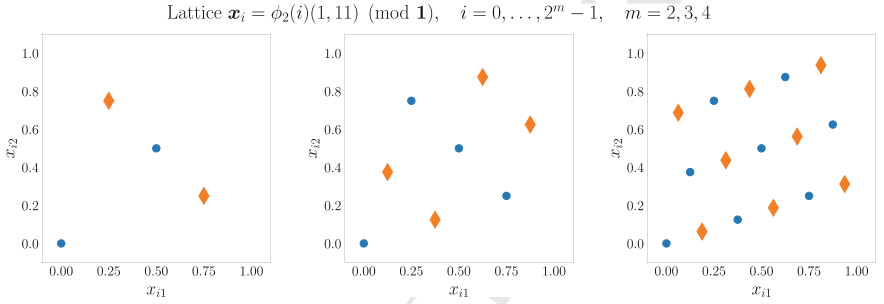


Fig. 6 An extensible lattice corresponding to Fig. 5 with the nodes reordered using the van der Corput sequence in base 2. For each plot the blue dots are a copy of the nodes to the left and the orange diamonds are a shifted copy (modulo $\mathbf{1}$) of the blue dots

of lattice nodes of size $n = b^m$ for a range of m [31]. The construction of great generating vectors for lattices has attracted a great deal of interest and is discussed further in Sect. 3.4.

3.2 *Digital Sequences*

Another family of LD sequences that generalize the van der Corput sequence (9) are digital sequences [18, 67]. For simplicity, we restrict ourselves to $b = 2$ and let $\oplus$ denote binary digitwise addition modulo 2 (also known as digitwise exclusive or), e.g., $3/8 \oplus 3/4 = {}_20.011 \oplus {}_20.110 = {}_20.101 = 5/8$. A digital sequence is defined as

$$\boldsymbol{x}_i^{\text{dig}} := i_0 \boldsymbol{x}_1^{\text{dig}} \oplus i_1 \boldsymbol{x}_2^{\text{dig}} \oplus i_2 \boldsymbol{x}_4^{\text{dig}} \oplus \cdots \in [0, 1)^d \quad \text{for } i = i_0 + i_1 2 + i_2 2^2 + \cdots, \tag{11}$$

where $\boldsymbol{x}_1^{\text{dig}}, \boldsymbol{x}_2^{\text{dig}}, \boldsymbol{x}_4^{\text{dig}}, \boldsymbol{x}_8^{\text{dig}}, \ldots$ are carefully chosen to ensure low discrepancy. (For $d = 1$ and $x_{2^m}^{\text{dig}} = 2^{-m-1}$, this is the van der Corput sequence in base 2.) The node set $\{\boldsymbol{x}_0^{\text{dig}}, \ldots, \boldsymbol{x}_{2^m-1}^{\text{dig}}\}$ for non-negative integer m is called a digital net and is closed under $\oplus$.

Traditionally, digital nets are defined in terms of $N \times M$ generating matrices, C_j, as follows:

$$\begin{pmatrix} x_{i,j,1}^{\text{dig}} \\ x_{i,j,2}^{\text{dig}} \\ \vdots \end{pmatrix} = \mathsf{C}_j \begin{pmatrix} i_0 \\ i_1 \\ \vdots \end{pmatrix} \pmod 2, \qquad j = 1, \ldots, d,\ i = 0, 1, \ldots, \tag{12}$$

where $x_{i,j,\ell}^{\text{dig}}$ denotes the ℓth digit of the jth component of the ith node. Here N is the maximum number of bits in the expression for $\boldsymbol{x}_i^{\text{dig}}$, say 52, and 2^M is the maximum number of nodes that is intended to be generated. It can be seen that definitions (11) and (12) are equivalent by noting that $\mathsf{C}_j = \left(x_{2^m,j,\ell}^{\text{dig}}\right)_{\ell,m=0}^{N-1,M-1}$.

To understand how well these nodes are evenly distributed over $[0, 1)^d$, imagine a box of the form

$$[a_1 b^{-k_1}, (a_1+1)2^{-k_1}) \times \cdots \times [a_d b^{-k_d}, (a_d+1)2^{-k_d}),$$
$$a_j \in \{0, \ldots, 2^{k_j} - 1\},\ k_j \in \{0, 1, \ldots\}, \qquad j = 1, \ldots d,\ \boldsymbol{k} \in \mathbb{N}_0. \tag{13}$$

This box has volume $2^{-(k_1+\cdots+k_d)} = 2^{-\|\boldsymbol{k}\|_1}$. For a digital net with 2^m nodes and $\|\boldsymbol{k}\|_1 \le m$, a "fair share" of nodes for this box would be $2^{m-\|\boldsymbol{k}\|_1}$. This will occur if and only if

$$\begin{aligned} &\text{the first } k_1 \text{ rows of the first } m \text{ columns of } \mathsf{C}_1 \text{ plus} \\ &\text{the first } k_2 \text{ rows of the first } m \text{ columns of } \mathsf{C}_2 \text{ plus} \\ &\qquad\qquad \vdots \\ &\text{the first } k_d \text{ rows of the first } m \text{ columns of } \mathsf{C}_d \\ &\text{are linearly independent over addition and multiplication modulo 2,} \end{aligned} \tag{14}$$

regardless of the choice of $\boldsymbol{a} = (a_1, \ldots, a_d)$. The t-value of a digital net with 2^m nodes is defined as any t for which condition (14) holds for all $\boldsymbol{k}$ with $\|\boldsymbol{k}\|_1 \le m - t$. Equivalently, for this t, every box of the form (13) with volume $2^{\|\boldsymbol{k}\|_1 - m}$ contains its fair share of 2^t nodes. Such a digital net is then called a (t, m, d)-net. An infinite sequence defined by (12) where every box of the form (13) contains its fair share of elements of the subsequence $\{\boldsymbol{x}_{p2^m}^{\text{dig}}, \ldots \boldsymbol{x}_{(p+1)2^m-1}^{\text{dig}}\}$ for all non-negative m and p is called a digital (t, d)-sequence. Ideally, one would like to construct (t, m, d)-nets and (t, d)-sequences with t as small as possible.

To illustrate the t-value, consider the following $d = 3$ dimensional digital net with 2^3 (eight) nodes, i.e., $m = 3$:

$$\{(0,0,0),(0.5,0.5,0.5),(0.25,0.75,0.75),(0.75,0.25,0.25),(0.125,0.625,0.375),\\ (0.625,0.125,0.875),(0.375,0.375,0.625),(0.875,0.875,0.125)\}. \quad (15)$$

Figure 7 shows several two dimensional projections of these nodes and the two-dimensional boxes (rectangles) of the form (13) with $\|\boldsymbol{k}\|_1 = 3$. Most of the boxes contain their fair share of one node, but the box $[0, 1/4) \times [0, 1) \times [0, 1/2)$ contains two nodes and the adjacent box, $[1/4, 1/2) \times [0, 1) \times [0, 1/2)$ contains no nodes. Thus, the t-value cannot be 0. However, the t-value is 1 because all boxes of the form (13) with $\|\boldsymbol{k}\|_1 = 2$, i.e., a volume of $2^{-2} = 1/4$ contain a fair share of $2^{m-\|\boldsymbol{k}\|_1} = 2$ nodes. The node set (15) is a $(1, 3, 3)$-net.

This conclusion can also be reached by looking at the first three rows and columns of the generating matrices for this digital net, which are

$$\mathsf{C}_1 = \begin{pmatrix} 1&0&0&\cdots\\ 0&1&0&\cdots\\ 0&0&1&\cdots\\ \vdots&\vdots&\vdots&\ddots \end{pmatrix}, \quad \mathsf{C}_2 = \begin{pmatrix} 1&1&1&\cdots\\ 0&1&0&\cdots\\ 0&0&1&\cdots\\ \vdots&\vdots&\vdots&\ddots \end{pmatrix}, \quad \mathsf{C}_3 = \begin{pmatrix} 1&1&0&\cdots\\ 0&1&1&\cdots\\ 0&0&1&\cdots\\ \vdots&\vdots&\vdots&\ddots \end{pmatrix}.$$

For $\boldsymbol{k} = (2, 0, 1)$ and $\|\boldsymbol{k}\|_1 = 3$, condition (14) is not satisfied, and so t cannot be 0 for the node set (15). However, condition (14) is satisfied for all $\boldsymbol{k}$ with $\|\boldsymbol{k}\|_1 = 2$, again confirming that we have a $(1, 3, 3)$-net.

If we consider only the first two coordinates of the node set defined in (15), then condition (14) is satisfied for all $\boldsymbol{k}$ with $\|\boldsymbol{k}\|_1 = 3$. The first two coordinates of (15) are a $(0, 3, 2)$-net.

Digital sequence generators can be found via number theory [18, Chap. 8] or numerical search (see [45] and [18, Chap. 10]). The earliest instance is due to Sobol' [83].

3.3 Halton Sequences

The Halton sequence is defined in terms of the van der Corput sequences for different bases:

$$\boldsymbol{x}_i^{\mathrm{Hal}} = \big(\phi_{b_1}(i), \ldots, \phi_{b_d}(i)\big), \qquad i = 0, 1, \ldots, \quad (16)$$

where $b_1, \ldots, b_d$ is a choice of d distinct co-prime numbers. Often they are chosen as the first d prime numbers. Halton sequences can be used with arbitrary sample sizes n, although its equidistribution properties are best when n is a product of powers of the d bases used. Figure 8 shows two dimensional projections of Halton nodes in six dimensions.

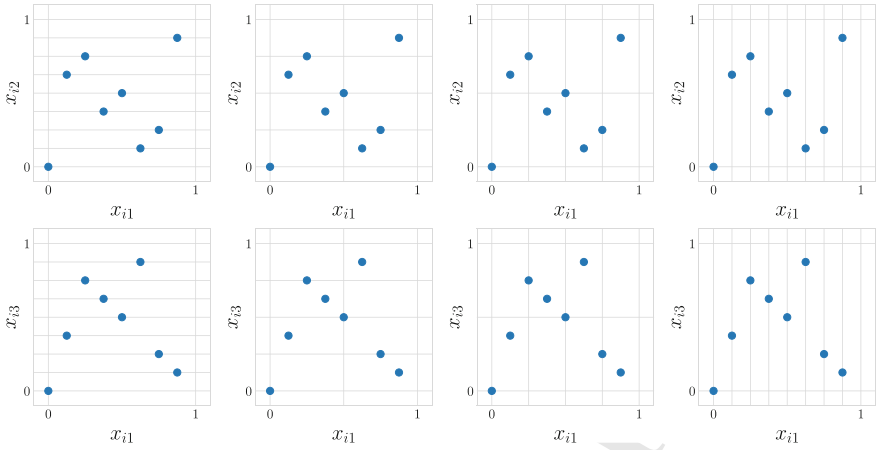


Fig. 7 Each box in the two dimensional projections of the node set (15) plotted above contain one node except for the second row, third plot from the left. Thus, this node set is not a (0, 3, 3)-net. It is however a (1, 3, 3)-net

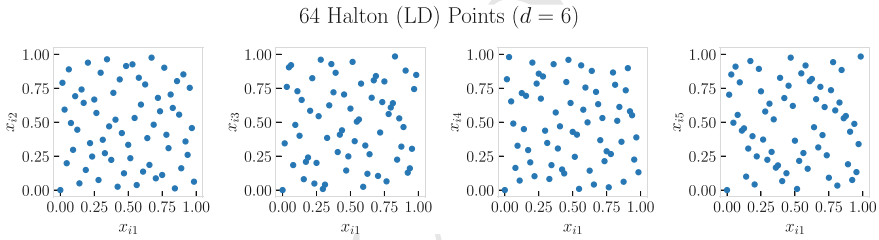


Fig. 8 Halton nodes as defined in (16) have low-discrepancy

The Halton construction is an extensible LD sequence. However, should one know $n \in \mathbb{N}$ in advance, a finite $d+1$-dimensional LD node set called the Hammersley node set can be defined similarly as

$$\boldsymbol{x}_i^{\text{Ham}} = \left(\frac{i}{n}, \phi_{b_1}(i), \ldots, \phi_{b_d}(i) \right) \quad i = 0, \ldots, n-1. \tag{17}$$

3.4 LD Nodes by Optimization

Most qMC constructions are studied based on their asymptotic behavior as $n \to \infty$, with numerous results on asymptotic rates [67, 68]. However, few methods address finding optimal node configurations for specific n and d which can be critical in practical applications where each function evaluation is quite costly. Of course, one possible strategy is to truncate asymptotically optimal sequences to the desired n,

however, as discussed above many of the standard LD constructions have a preferred number of nodes with truncations leading to imbalances in equidistribution. We can often do better optimizing directly for the target n and d.

When optimizing LD nodes, one requires an objective function which in some sense must assess the quality of the LD nodes directly, or the expected performance when the node set is implemented in a qMC estimate (3). We will refer to such an objective function as a figure-of-merit (FOM) and popular choices include the discrepancy (see Sect. 5) or worst-case error metrics when restricted to a specific space of functions; see [15, 30].

For several decades, combinatorial optimization techniques have been employed to obtain good generating parameters for LD constructions. For example, in the case of a lattice rule which is fully determined by the generating vector $\boldsymbol{h} \in \mathbb{Z}_n^d := \{0, 1, \ldots, n-1\}^d$ given n and d, one often resorts to finding good $\boldsymbol{h}$ via computer search to minimize a chosen FOM. An exhaustive search is infeasible due to the exponential growth of the search space with d.

Thus, the component-by-component (CBC) construction, introduced by Korobov [53] and later revisited by Sloan and collaborators [80], reduces the search space to size dn by constructing $\boldsymbol{h}$ one component at a time while keeping the previous components fixed. This greedy procedure also applies to finding good generating matrices C_j for digital sequences. For certain choices of function spaces, there exist fast component-by-component construction implementations using fast Fourier transforms (FFTs) as the primary tool. We refer to [16, 17, 44, 54] and references therein for works on the CBC construction for lattice rules and digital sequences, [69, 70] for their fast implementations and [15, 58] for a qMC software tool `LatNet Builder` by L'Ecuyer to generate LD nodes which relies heavily on the CBC construction method.

When constructing LD nodes, it is essential to ensure not only a well-distributed set of nodes in the d-dimensional space but also in its lower-dimensional projections (recall the Cartesian grid from Sect. 1, Fig. 2). `MatBuilder` [76] is a software tool developed precisely for this task; to generate generating matrices for digital sequence LD nodes possessing excellent low-dimensional uniformity. In this software, the solutions to integer linear programming problems are used to build the columns of generator matrices in a greedy manner for the LD digital sequence.

We have outlined various efforts by qMC researchers to optimize specific classes of LD constructions, including lattices and digital sequences. Significant effort has even been taken to optimize the uniformity of so-called generalized Halton sequences due to the poor lower dimensional projections exhibited by even a moderate dimensional Halton sequence [19, 51]. These methods are discussed briefly in Sect. 4.3. There also exist several efforts to generate novel LD constructions which do not adhere to any underlying number theoretic structure, but none-the-less, possess a high level of uniformity. These methods are often global optimization methods of sample point distributions, which is often challenging due to the non-convex nature of the FOM as a function of the nodes. One of the first works to globally optimize LD nodes, albeit a heuristic approach, was by Winker and Fang [92] employing the

threshold accepting optimization algorithm to directly optimize the uniformity of the LD nodes with respect to a discrepancy FOM.

More recently, Rusch, Kirk, Bronstein, Lemieux and Rus [78] developed the first machine-learning assisted optimization tool for LD node optimization in the Message-Passing Monte Carlo (MPMC) method. MPMC implements tools from geometric deep learning, imposing a computational graph onto an initialized IID node set to transform the nodes to LD node set via a learned mapping. The optimization is gradient-based and is guided by the direct minimization of a discrepancy-based FOM.

We also highlight the method proposed by Hinrichs and Oettershagen [38] generating optimal LD nodes for a specific class of functions in two-dimensions. To address the non-convexity challenge, the authors decompose the global optimization of a specific FOM into a large number of smaller, convex sub-problems which are solved individually.

There are several other recent attempts to construct optimal LD nodes, many of which are well summarized in the PhD thesis of Clément [11]. We highlight the subset selection method proposed in [13] to choose from an n-element initialization set, the $k < n$ nodes which minimize a discrepancy FOM. A heuristic approach to this problem was later shown in [14]. Furthermore, a method to generate optimal star-discrepancy node sets for fixed n and d based on a non-linear programming approach was suggested in [12], however is computationally restricted to very small numbers of nodes in dimensions two and three only.

4 Randomization

The LD constructions described in the previous section have so far been *deterministic*. The sample mean, $\hat{\mu}_n$, does not change each time it is computed like it would for IID nodes. While determinism has advantages, randomization has substantial advantages as well.

- Randomization done correctly removes bias in the estimator, $\hat{\mu}_n$.
- Replications of random estimators can facilitate error estimates for the sample mean.
- Often the application of interest requires transforming the LD nodes to mimic other distributions. This is the case in the Keister example of Sect. 1, as seen in (6) where the nodes, $\{\boldsymbol{x}_i\}_{i=0}^{n-1}$, are transformed to mimic a Gaussian distribution. The LD sequences defined in the previous section start with $\boldsymbol{x}_0 = \mathbf{0}$, as can be seen in (8), (11), (16), and Figures 5, 6, 7 and 8. The node $\mathbf{0}$ becomes infinite under a transformation to mimic a Gaussian distribution, which can trigger runtime errors. Randomization eliminates nodes on the boundary of the unit cube.

The key to good randomization is to preserve the LD quality of the node sequence. In general, we recommend randomization, if available.

4.1 Shifts

The simplest randomization is to shift the node sequence by $\boldsymbol{\Delta} \sim \mathcal{U}[0,1)^d$. For lattice sequences the shift is applied modulo $\mathbf{1}$ so that shifted extensible lattice turns (10) into

$$\boldsymbol{x}_i^{\boldsymbol{\Delta}\text{-lat}} := \boldsymbol{x}_i^{\text{lat}} + \boldsymbol{\Delta} \pmod{\mathbf{1}} = \phi_b(i)\boldsymbol{h} + \boldsymbol{\Delta} \pmod{\mathbf{1}}, \qquad i = 0, 1, \ldots. \tag{18}$$

Since the first 2^m elements of the lattice sequence form a group under modulo $\mathbf{1}$ addition, the corresponding shifted node set is a coset. Figure 9 illustrates three shifts of the lattice plotted in Fig. 5.

For digital sequences the shift should be applied using digitwise addition. A digital shift of the digital sequence defined in (11) then becomes

$$\boldsymbol{x}_i^{\boldsymbol{\Delta}\text{-dig}} := \boldsymbol{x}_i^{\text{dig}} \oplus \boldsymbol{\Delta} = i_0\boldsymbol{x}_1^{\text{dig}} \oplus i_1\boldsymbol{x}_2^{\text{dig}} \oplus i_2\boldsymbol{x}_4^{\text{dig}} \oplus \cdots \oplus \boldsymbol{\Delta} \quad \text{for } i = i_0 + i_1 2 + i_2 2^2 + \cdots. \tag{19}$$

Three digital shifts of the same net are given in Fig. 10. A digital shift does not alter the t-value of a (t, m, d)-net.

Let $\{\boldsymbol{x}_i^{\boldsymbol{\Delta}}\}_{i=0}^{n-1}$ denote a uniform (digital or modulo $\mathbf{1}$) random shift of any deterministic original node set, $\{\boldsymbol{x}_i\}_{i=0}^{n-1}$. That is, $\boldsymbol{x}_i^{\boldsymbol{\Delta}} := \boldsymbol{x}_i + \boldsymbol{\Delta} \pmod{\mathbf{1}}$ or $\boldsymbol{x}_i^{\boldsymbol{\Delta}} := \boldsymbol{x}_i \oplus \boldsymbol{\Delta}$ and $\boldsymbol{\Delta} \sim \mathcal{U}[0,1)^d$. This then implies that each $\boldsymbol{x}_i^{\boldsymbol{\Delta}} \sim \mathcal{U}[0,1)^d$ and thus $\hat{\mu}_n$ is an unbiased estimator of μ, which was mentioned as an advantage at the beginning of this section.

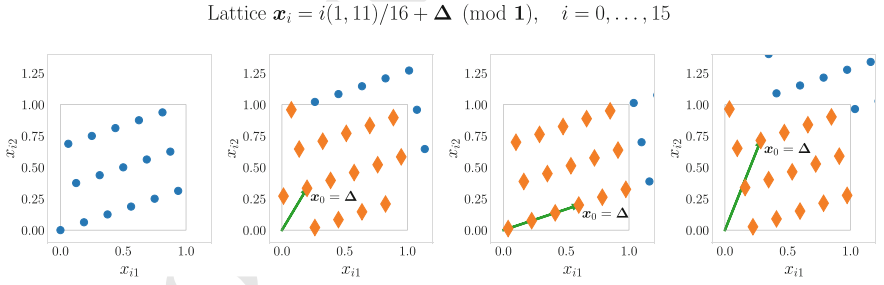


Fig. 9 The original unshifted lattice (left) and three different shifts modulo $\mathbf{1}$ of the lattice. Note that the structure of the lattice is maintained

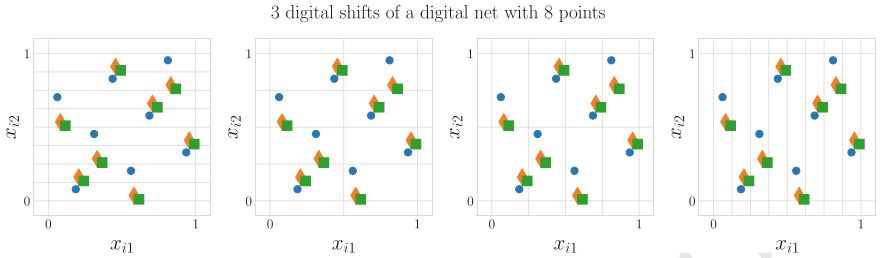


Fig. 10 Three digital shifts of a digital $(0, 3, 2)$-net with nodes denoted by three different colors/shapes. There is one node in each smaller box of the form (13) for $\|\boldsymbol{k}\| = 3$, for each digitally shifted net

4.2 Digital Scrambles

For digital nets one can randomize even further by a scrambling that preserves the t-value. Owen [72] proposed a scrambling of nets called nested uniform scrambling. A simpler version called linear matrix scrambling was proposed by Matoušek [65] and is described here. Starting from the formulation of digital sequences involving generator matrices in (12), we multiply each generator matrix on the left by a lower triangular matrix with ones along the diagonals and elements below the diagonal that are randomly chosen to be 0 or 1 with equal probability:

$$\begin{pmatrix} x_{ij1}^{\text{scr-net}} \\ x_{ij2}^{\text{scr-net}} \\ \vdots \end{pmatrix} = \mathsf{L_j C}_j \begin{pmatrix} i_0 \\ i_1 \\ \vdots \end{pmatrix} + \begin{pmatrix} \Delta_{1j} \\ \Delta_{2j} \\ \vdots \end{pmatrix} \pmod 2, \qquad j = 1, \ldots, d, \ i = 0, 1, \ldots, \tag{20a}$$

$$\mathsf{L_j} := \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots \\ l_{21} & 1 & 0 & 0 & \cdots \\ l_{31} & l_{32} & 1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \ddots \end{pmatrix}, \qquad l_{\ell j} \overset{\text{IID}}{\sim} \mathcal{U}\{0, 1\} \tag{20b}$$

A random digital shift, $\boldsymbol{\Delta}$ is also added, where $\Delta_{\ell j}$ denotes the ℓ^{th} digit of the j^{th} dimension of the shift. Digital scrambling of a (t, m, d)-net does not alter its t-value (Fig. 11).

4.3 Randomizations for Halton

As Halton nodes can be viewed as digital nets with different bases for each dimension, the randomization techniques used for digital nets can be applied to Halton node sets. Random nested uniform scrambling and linear matrix scrambling have also been considered for Halton node sets [74].

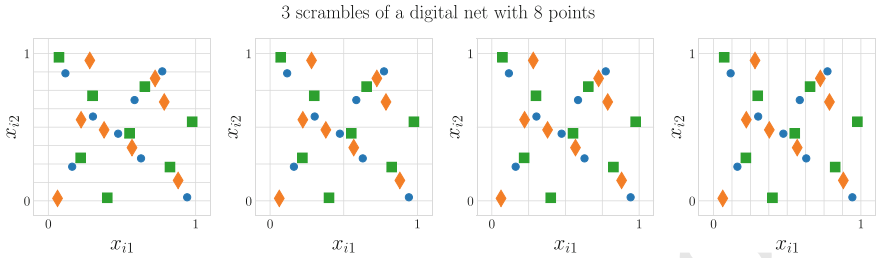


Fig. 11 Three linear scrambles of a digital $(0, 3, 2)$-net with nodes denoted by three different colors/shapes. There is one node in each smaller box of the form (13) for $\|\boldsymbol{k}\| = 3$, for each digitally shifted net

We detail one possible randomization of the Halton sequence which is equivalent to applying a random digital shift (19) in the respective base of each dimension of the Halton sequence. We first define the generalized van der Corput sequence in base b by obtaining a sequence $\Sigma = (\sigma_r)_{r\ge 0}$ of permutations of $\{0, \dots, b-1\}$ where the i^{th} term of the one-dimensional sequence is defined as

$$\phi_b^{\Sigma}(i_0 + i_1 b + i_2 b^2 + \cdots) := \sigma_0(i_0) b^{-1} + \sigma_1(i_1) b^{-2} + \sigma_2(i_2) b^{-3} + \cdots \in [0, 1)$$
$$\text{where } i_0, \sigma_0(i_0), i_1, \sigma_1(i_1), \dots \in \{0, \dots b-1\}. \quad (21)$$

The generalized Halton sequence is subsequently defined by choosing d sequences of permutations $\Sigma_j = (\sigma_{j,r})_{r\ge 0}$ of $\{0, \dots, b_j - 1\}$ for $j = 1, 2, \dots, d$ and setting

$$\boldsymbol{x}_i^{\text{G-Hal}} = \left(\phi_{b_1}^{\Sigma_1}(i), \dots, \phi_{b_d}^{\Sigma_d}(i)\right), \quad i = 0, 1, \dots.$$

A randomized qMC estimate $\hat{\mu}_n$ is then an unbiased estimator of μ if the permutations Σ_j are sampled independently and uniformly, i.e μ if $\Sigma_j \overset{\text{IID}}{\sim} \mathcal{U}\{0, \dots, b_j - 1\}^d$ since this ensures $\boldsymbol{x}_i^{\text{G-Hal}} \sim \mathcal{U}[0, 1)^d$.

We note in passing that many works [3, 9, 86, 87] have sought to optimize the permutations Σ_j deterministically to enhance uniformity. For a comprehensive survey of the Halton sequence (up to 2008), we refer readers to the excellent work by Faure and Lemieux [19].

4.4 Randomized Generators

Instead of explicitly searching for good generating vectors or matrices, one can instead sample a few random generating vectors or generating matrices and then prune out bad choices. One recent method has explored selecting the median of r randomized qMC estimates, rather than the mean [75]. Random generating vectors

for lattices take $\boldsymbol{h} \overset{\text{IID}}{\sim} \mathcal{U}\{1, \ldots, n-1\}^d$ while random generating matrices for base 2 digital nets—as described in (12)—set $x_{ij\ell}^{\text{dig}} \overset{\text{IID}}{\sim} \mathcal{U}\{0, 1\}$ for $i < \ell$ while enforcing $x_{iji} = 1$ and $x_{ij\ell} = 0$ for $i > \ell$.

5 Discrepancy

So far, we have relied on eye tests and the Keister example to show how LD sequences are better than IID. This section introduces the theory that shows why minimizing discrepancy leads to better approximations to μ.

5.1 Discrepancies Defined by Kernels

Let $K : [0, 1)^d \times [0, 1)^d \to \mathbb{R}$ be a function satisfying the following two conditions:

$$\text{Symmetry:} \quad K(\boldsymbol{t}, \boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{t}) \qquad \forall \boldsymbol{t}, \boldsymbol{x} \in [0, 1)^d, \tag{22a}$$

$$\text{Postitive definiteness:} \quad \sum_{i,j=0}^{n-1} c_i K(\boldsymbol{x}_i, \boldsymbol{x}_j) c_j > 0 \qquad \forall n \in \mathbb{N},\ \boldsymbol{c} \neq 0,$$

$$\text{distinct } \boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1} \in [0, 1)^d. \tag{22b}$$

Then it is known [2] that this K is the reproducing kernel for a Hilbert space, $\mathcal{H}_K$ with associated inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}_K}$, such that

$$\text{Belonging:} \quad K(\cdot, \boldsymbol{x}) \in \mathcal{H}_K \qquad \forall \boldsymbol{x} \in [0, 1)^d, \tag{23a}$$

$$\text{Reproducing:} \quad f(\boldsymbol{x}) = \langle K(\cdot, \boldsymbol{x}), f \rangle \qquad \forall \boldsymbol{x} \in [0, 1)^d,\ f \in \mathcal{H}_K. \tag{23b}$$

Having a reproducing kernel Hilbert space with a known kernel K allows us to derive a rigorous error bound for $\mu - \hat{\mu}_n$. First note that for any bounded, linear functional, L on the Hilbert space $\mathcal{H}_K$, there is some $\eta_L \in \mathcal{H}_K$ such that

$$L(f) = \langle \eta_L, f \rangle_{\mathcal{H}_K} \qquad \forall f \in \mathcal{H}_K.$$

This is guaranteed by the Riesz Representation Theorem, and η_L is called the representer of L. Using the reproducing property of K, one may derive an explicit formula for $\eta_L(\boldsymbol{x})$, namely,

$$\eta_L(\boldsymbol{x}) = \langle K(\cdot, \boldsymbol{x}), \eta_L \rangle_{\mathcal{H}_K} = L\big(K(\cdot, \boldsymbol{x})\big).$$

From this expression for $\eta_L(\boldsymbol{x})$ one may then calculate the squared norm of the linear functional L as the squared norm of its representer

$$\|\eta_L\|_{\mathcal{H}_K}^2 = \langle \eta_L, \eta_L \rangle_{\mathcal{H}_K} = L(\eta_L) = L^{\cdot\cdot}\Big(L^{\cdot}\big(K(\cdot,\cdot\cdot)\big)\Big) \tag{24}$$

where $L^{\cdot}$ denotes applying linear functional L to the first argument of the kernel K, and $L^{\cdot\cdot}$ indicates applying L to the resulting function of the second variable only.

If $\mu(\cdot) : \mathcal{H}_K \to \mathbb{R}$ is a bounded linear functional on $\mathcal{H}_K$, we may use the argument above to derive our error bound. First we write the dependence of $\mu - \hat{\mu}_n$ on f explicitly as $\mu(f) - \hat{\mu}_n(f)$ and note that the error functional, $\mu(\cdot) - \hat{\mu}_n(\cdot)$, is linear. The Riesz Representation Theorem and the Cauchy-Schwarz inequality imply a tight error bound

$$\begin{aligned} |\mu(f) - \hat{\mu}_n(f)| &= |\langle \eta_{\mu(\cdot)-\hat{\mu}_n(\cdot)}, f \rangle_{\mathcal{H}_K}| \le \|\eta_{\mu(\cdot)-\hat{\mu}_n(\cdot)}\|_{\mathcal{H}_K} \|f\|_{\mathcal{H}_K} \\ |\mu(f) - \hat{\mu}_n(f)| &\le \underbrace{\|\eta_{\mu(\cdot)-\hat{\mu}_n(\cdot)}\|_{\mathcal{H}_K}}_{=:\text{discrepancy}\left(\{\boldsymbol{x}\}_{i=0}^{n-1}, K\right)} \underbrace{\inf_{c\in\mathbb{R}} \|f - c\|_{\mathcal{H}_K}}_{=:\text{variation}(f,K)} \qquad \forall f \in \mathcal{H}_K, \end{aligned} \tag{25}$$

since $\hat{\mu}_n$ is exact for constants. (It is assumed that $\mathcal{H}_K$ contains constant functions so that $f \in \mathcal{H}_K$ implies that $f - c \in \mathcal{H}_K$.) The squared norm of the error funtional can be expressed explicitly via (24) as

$$\begin{aligned} \text{discrepancy}^2\left(\{\boldsymbol{x}\}_{i=0}^{n-1}, K\right) &= \|\eta_{\mu(\cdot)-\hat{\mu}_n(\cdot)}\|_{\mathcal{H}_K}^2 \\ &= \big(\mu(\cdot\cdot) - \hat{\mu}_n(\cdot\cdot)\big)\Big(\big(\mu(\cdot) - \hat{\mu}_n(\cdot)\big)\big(K(\cdot,\cdot\cdot)\big)\Big) \\ &= \int_{[0,1)^d\times[0,1)^d} K(\boldsymbol{t},\boldsymbol{x})\,\mathrm{d}\boldsymbol{t}\,\mathrm{d}\boldsymbol{x} - \frac{2}{n}\sum_{i=0}^{n-1}\int_{[0,1)^d} K(\boldsymbol{t},\boldsymbol{x}_i)\,\mathrm{d}\boldsymbol{t} \\ &\quad + \frac{1}{n^2}\sum_{i,j=0}^{n-1} K(\boldsymbol{x}_i,\boldsymbol{x}_j), \end{aligned} \tag{26}$$

where again the $(\cdot)$ and $(\cdot\cdot)$ notations represent the functional acting on the first and second arguments, respectively.

The error bound (25) has two factors:

- The discrepancy, which is the norm of the error functional, depends only on the node set, $\{\boldsymbol{x}\}_{i=0}^{n-1}$, and *not* on f. The discrepancy measures the deficiency of that node set.
- The variation depends only on f, the function defining our random variable Y whose expectation we wish to compute, and *not* on the node set, $\{\boldsymbol{x}\}_{i=0}^{n-1}$. The variation is a measure of the roughness of f.

In general, designers of qMC methods want to construct sets or sequences of nodes with discrepancy as small as possible, either by increasing n, if the computational

budget allows, or by better placement of the nodes. The constructions described in Sect. 3 are ways of ensuring better placement of the nodes, provided that the generators are chosen well. Practitioners of qMC want to formulate μ in a way to make the variation as small as possible. This is discussed in Sect. 7.

Although error bound (25) is elegant, it leaves several matters unresolved:

- The reproducing kernel K defines both the discrepancy and the variation. This means that different choices of K will lead to different error bounds for the same $\hat{\mu}_n$, even though the error is unchanged.
- The Hilbert space $\mathcal{H}_K$ contains one's assumptions about f, such as smoothness or periodicity. Knowing K, however, does not automatically lead to an explicit formula for the inner product of the associated Hilbert space, $\langle \cdot, \cdot \rangle_{\mathcal{H}_K}$. Conversely, having an explicit formula for $\langle \cdot, \cdot \rangle_{\mathcal{H}_K}$ does not automatically lead to an explicit formula for the reproducing kernel, K. In both cases, educated guesswork is involved.
- Choosing a more restrictive $\mathcal{H}_K$ may lead to a sharper error bound, however, it is often infeasible in practice to check whether the f lies in $\mathcal{H}_K$. This is what we mean in the introduction by referring to f as a "black box".
- Moreover, even if one is confident that $f \in \mathcal{H}_K$ (f is a "gray box"), it is usually impractical to compute variation(f, K), as we shall see in the discrepancy example below. Error bound (25) cannot be used as a stopping criterion to determine the n needed to satisfy an error tolerance.

In spite of these drawbacks, (25) is quite useful:

- There are families of commonly used reproducing kernels [5, 27–29]. Once K has been fixed, the discrepancy may be used to compare different LD sequences. The discrepancy's rate of decay—which sometimes can be computed theoretically—indicates how quickly $\hat{\mu}_n$ approaches μ.
- When an explicit formula for the inner product of the associated Hilbert space, $\langle \cdot, \cdot \rangle_{\mathcal{H}_K}$, can be deduced, it suggests how to formulate μ to reduce variation(f, K).

5.2 The Centered Discrepancy

An example of a reproducing kernel defined on $[0, 1)^d \times [0, 1)^d$ is

$$K(\boldsymbol{t}, \boldsymbol{x}) = \prod_{\ell=1}^{d} \left[1 + \frac{1}{2} \big(|t_\ell - 1/2| + |x_\ell - 1/2| - |t_\ell - x_\ell| \big) \right]. \tag{27}$$

One can verify that this K satisfies the symmetry condition (22a). It also is symmetric about the middle of the unit cube, i.e.,

$$K\big((t_1, \ldots, t_{\ell-1}, 1 - t_\ell, t_{\ell+1}, \ldots, t_d), (x_1, \ldots, x_{\ell-1}, 1 - x_\ell, x_{\ell+1}, \ldots, x_d)\big) = K(\boldsymbol{t}, \boldsymbol{x}),$$

which is a desirable property for problems where there is no preferred corner of the unit cube.

The inner product for the Hilbert space $\mathcal{H}_K$ with reproducing kernel K is

$$\begin{aligned}
\langle f, g\rangle_{\mathcal{H}_K} &= f(1/2,\dots,1/2)g(1/2,\dots,1/2) \\
&\quad + \int_{[0,1)} [D^{\{1\}} f](x_1)\,[D^{\{1\}} g](x_1)\,\mathrm{d}x_1 + \int_{[0,1)} [D^{\{2\}} f](x_2)\,[D^{\{2\}} g](x_2)\,\mathrm{d}x_2 \\
&\qquad + \cdots \\
&\quad + \int_{[0,1)^2} [D^{\{1,2\}} f](x_1,x_2)\,[D^{\{1,2\}} g](x_1,x_2)\,\mathrm{d}x_1\mathrm{d}x_2 + \cdots \\
&\qquad + \cdots + \int_{[0,1)^d} [D^{\{1,\dots,d\}} f](\boldsymbol{x})\,[D^{\{1,\dots,d\}} g](\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \\
&= \sum_{\boldsymbol{\ell}\subseteq\{1,\dots,d\}} \langle D^{\boldsymbol{\ell}} f, D^{\boldsymbol{\ell}} g\rangle_2 \qquad (28)
\end{aligned}$$

where $\langle\cdot,\cdot\rangle_2$ denotes the L^2 inner product, the operator $D^{\boldsymbol{\ell}}$ is defined as

$$\begin{aligned}
&[D^{\{\ell_1,\ell_2,\dots,\ell_s\}} f](x_1,x_2,\dots,x_s) \\
&:= \frac{\partial^s f(1/2,\dots,1/2,x_{\ell_1},1/2,\dots,1/2,x_{\ell_2},1/2,\dots,1/2,x_{\ell_s},1/2,\dots,1/2)}{\partial x_{\ell_1}\partial x_{\ell_2}\cdots\partial x_{\ell_s}},
\end{aligned}$$

and $D^{\emptyset} f$ is understood to be the constant $f(1/2,\dots,1/2)$ [27]. The variation, which corresponds to the non-constant part of the function, is

$$\mathrm{variation}(f) := \sqrt{\sum_{\emptyset\neq\boldsymbol{\ell}\subseteq\{1,\dots,d\}} \|D^{\boldsymbol{\ell}} f\|_2^2}.$$

We show that K defined in (27) is the reproducing kernel for the $\mathcal{H}_K$ with the above inner product for $d = 1$ and assuming $0 \le x \le 1/2$:

$$\begin{aligned}
\langle K(\cdot,x), f\rangle_{\mathcal{H}_K} &= K(1/2,x)f(1/2) + \int_0^1 \frac{\partial K(t,x)}{\partial t} f'(t)\,\mathrm{d}t \\
&= 1\times f(1/2) + \int_0^x \frac{1}{2}[-1-(-1)]f'(t)\,\mathrm{d}t + \int_x^{1/2} \frac{1}{2}[-1-1]f'(t)\,\mathrm{d}t \\
&\quad + \int_{1/2}^1 \frac{1}{2}[1-1]f'(t)\,\mathrm{d}t \\
&= f(1/2) + 0 - \int_x^{1/2} f'(t)\,\mathrm{d}t + 0 = f(1/2) - [f(1/2) - f(x)] \\
&= f(x).
\end{aligned}$$

The extensions to general $1/2 < x < 1$ and to $d > 1$ are straightforward [27].

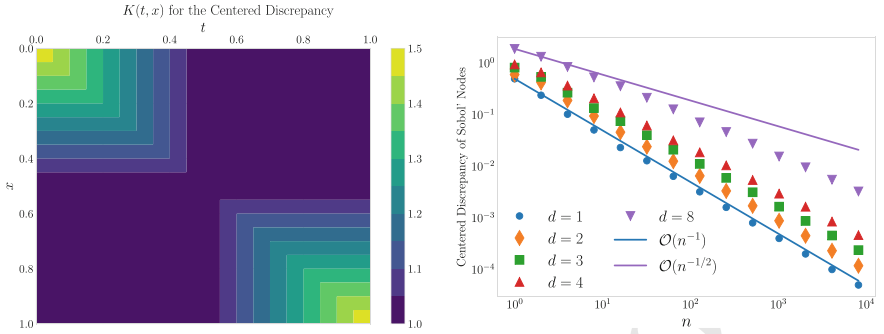


Fig. 12 Left: A plot of the reproducing kernel for the centered discrepancy defined in (27) for $d = 1$. Note that the vertical axis is inverted so that the plot is a picture of the "matrix" $\big(K(t,x)\big)_{t,x\in[0,1)]}$. Any submatrix is positive definite by property (22b). Right: The centered discrepancy for several modest values of d. They show an asymptotic decay close to $O(n^{-1})$, but this rate is achieved only at higher n for larger values of d

Figure 12 (left) is a color map of the reproducing kernel, K, defined in (27) for $d = 1$. As shown in the figure, the matrix $\mathsf{K} := \big(K(x_i, x_j)\big)_{i,j=1}^{n}$ that arises in condition (22b) is formed by taking n rows and corresponding columns of this plot. Any such matrix K is strictly positive definite, as is suggested by the higher values near the diagonal.

The squared centered discrepancy is given by substituting the formula for K into (26), which implies

$$\begin{aligned}&\text{ctr-discrepancy}^2\big(\{\boldsymbol{x}_i\}_{i=0}^{n-1}\big)\\ &\quad=\left(\frac{13}{12}\right)^d-\frac{2}{n}\sum_{i=0}^{n-1}\prod_{\ell=1}^{d}\left[1+\frac{1}{2}\big(|x_{i\ell}-1/2|-|x_{i\ell}-1/2|^2\big)\right]\\ &\quad\frac{1}{n^2}\sum_{i,j=0}^{n-1}\prod_{\ell=1}^{d}\left[1+\frac{1}{2}\big(|x_{i\ell}-1/2|+|x_{j\ell}-1/2|-|x_{i\ell}-x_{j\ell}|\big)\right].\end{aligned}\tag{29}$$

As is true for most cases, the computational cost of computing the discrepancy is $O(dn^2)$ as d and/or n tend to infinity.

Figure 12 (right) displays the discrepancy for a Sobol' LD digital net for several modest values of d and for $n = 1, 2, 4, 8, 16, \ldots$. The asymptotic decay rate of nearly $O(n^{-1})$ is observed for small d and can be anticipated for a bit larger d. Thus, the decay of the discrepancy mirrors the convergence rate for LD sequences for the Keister example in Sect. 1 and Fig. 1.

The centered discrepancy also has a geometric interpretation, as is described in [27]. Other choices of kernels and their geometric interpretations are explained there as well. Moreover, it is possible to define discrepancies for Banach spaces of integrands, however, the formulas for these discrepancies are not as amenable to calculation.

5.3 Coordinate Weights

When the dimension, d, is increased, the discrepancy increases and the rate of decay with increasing sample size deteriorates, as can be seen in Fig. 13 (top). This raises the question of whether the LD sequences lose their effectiveness for higher dimensions or whether the theory of the previous subsection does not coincide with practice.

Note that the discrepancy of the null set, which is equivalent to the norm of the linear functional $\mu(\cdot)$, is

$$\text{discrepancy}(\emptyset, K) = \sqrt{\int_{[0,1)^d \times [0,1)^d} K(\boldsymbol{t}, \boldsymbol{x}) \, \mathrm{d}\boldsymbol{t} \, \mathrm{d}\boldsymbol{x}}, \tag{30}$$

which corresponds to $(13/12)^{d/2}$ for the centered discrepancy. This means that integration gets exponentially harder as d increases. One may scale the discrepancy of a node set by dividing it by discrepancy$(\emptyset, K)$, but this does not recover the desired nearly $O(n^{-1})$ decay rate.

A good approach is to introduce coordinate weights, $\gamma_1, \ldots, \gamma_d$ into the reproducing kernel. These were introduced in [26] but begin to be fully exploited in [82] and the articles that followed. For the kernel defining the centered discrepancy, this means

$$K_{\boldsymbol{\gamma}}(\boldsymbol{t}, \boldsymbol{x}) = \prod_{\ell=1}^{d} \left[1 + \frac{\gamma_\ell^2}{2} \bigl(|t_\ell - 1/2| + |x_\ell - 1/2| - |t_\ell - x_\ell| \bigr) \right], \tag{31}$$

$$\langle f, g \rangle_{\mathcal{H}_{K_{\boldsymbol{\gamma}}}} = \sum_{\boldsymbol{\ell} \subseteq \{1,\ldots,d\}} \left(\prod_{k} \gamma_{\ell_k}^2 \right) \langle D^{\boldsymbol{\ell}} f, D^{\boldsymbol{\ell}} g \rangle_2, \tag{32}$$

$$\text{variation}(f, K_{\boldsymbol{\gamma}}) = \sqrt{\sum_{\boldsymbol{\ell} \subseteq \{1,\ldots,d\}} \left(\prod_{k} \gamma_{\ell_k}^{-2} \right) \| D^{\boldsymbol{\ell}} f \|_2^2}, \tag{33}$$

$$\begin{aligned} &\text{wt-ctr-discrepancy}^2\bigl(\{\boldsymbol{x}_i\}_{i=0}^{n-1}, \boldsymbol{\gamma}\bigr) \\ &= \prod_{\ell=1}^{d} \left(1 + \frac{\gamma_\ell^2}{12} \right)^d - \frac{2}{n} \sum_{i=0}^{n-1} \prod_{\ell=1}^{d} \left[1 + \frac{\gamma_\ell^2}{2} \bigl(|x_{i\ell} - 1/2| - |x_{i\ell} - 1/2|^d \bigr) \right] \\ &\quad + \frac{1}{n^2} \sum_{i,j=0}^{n-1} \prod_{\ell=1}^{d} \left[1 + \frac{\gamma_\ell^2}{2} \bigl(|x_{i\ell} - 1/2| + |x_{j\ell} - 1/2| - |x_{i\ell} - x_{j\ell}| \bigr) \right]. \end{aligned} \tag{34}$$

Figure 13 (bottom) plots the scaled centered discrepancy with coordinate weights $\boldsymbol{\gamma} = (1/\ell)_{\ell=1}^d$, i.e.,

$$\text{wt-ctr-discrepancy}\bigl(\{\boldsymbol{x}_i\}_{i=0}^{n-1}, \boldsymbol{\gamma}\bigr) / \text{wt-ctr-discrepancy}(\emptyset, \boldsymbol{\gamma}),$$

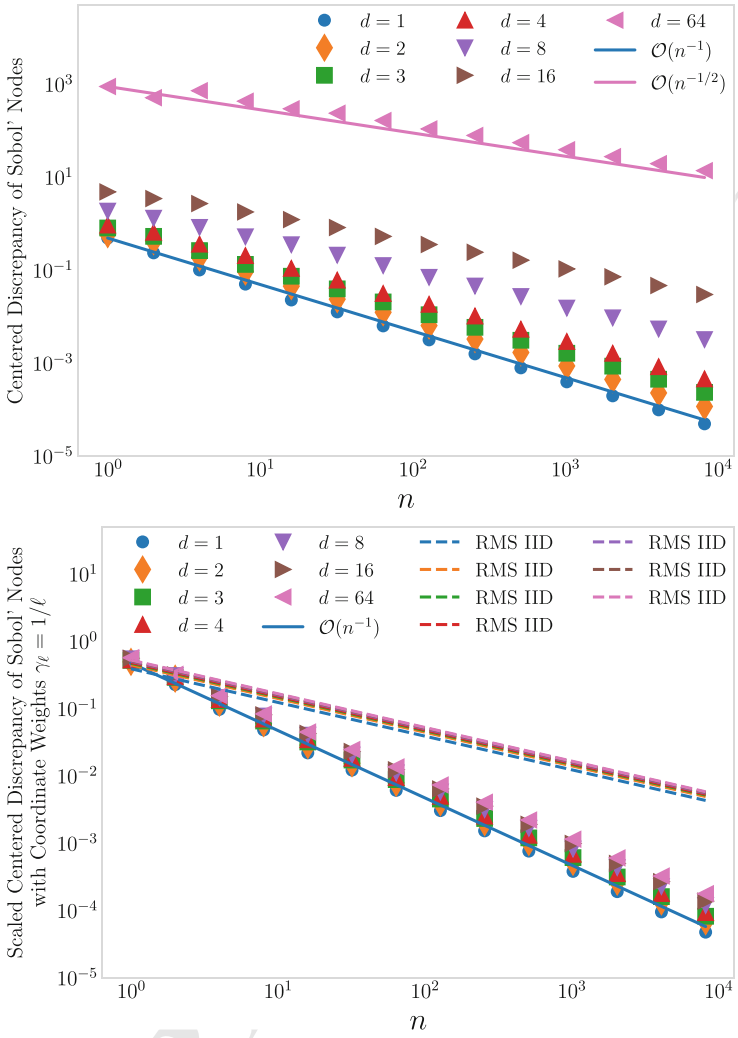


Fig. 13 Top: The centered discrepancy for a Sobol' LD sequence for a range of d. For larger d only a decay rate of $\mathcal{O}(n^{-1/2})$, is achieved for the range of n shown. Moreover, the discrepancy for larger d is huge. Bottom: The scaled centered discrepancy with coordinate weights for a Sobol' LD sequence for various d and the root mean squared scaled centered discrepancy with coordinate weights for IID nodes. LD nodes have a much smaller discrepancy than IID nodes

610417_1_En_1_Chapter-online ☑ TYPESET ☐ DISK ☐ LE ☑ CP Disp.:13/12/2025 Pages: 37 Layout: T1-Standard

against sample size for a range of d. Now, even for larger d there is a reasonable decay that approximates $O(n^{-1})$.

For IID samples the root mean squared discrepancy for a general reproducing kernel, K, is

$$\sqrt{\mathbb{E}[\text{discrepancy}^2(\{\boldsymbol{x}\}_{i=0}^{n-1}, K)]} = \frac{1}{\sqrt{n}}\sqrt{\int_{[0,1)^d} K(\boldsymbol{x},\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} - \int_{[0,1)^d\times[0,1)^d} K(\boldsymbol{t},\boldsymbol{x})\,\mathrm{d}\boldsymbol{t}\,\mathrm{d}\boldsymbol{x}}. \quad (35)$$

This is also plotted in Fig. 13 (right). which highlights the superiority of LD nodes.

The meaning of the coordinate weights is that those coordinates with a small weight should not contribute much to the variation. In (33), if the $\gamma_{\ell_1},\dots,\gamma_{\ell_s}$ are small, then $\prod_k \gamma_{\ell_k}^{-2}$ will be large. For f to have modest variation requires $\|D^{\boldsymbol{\ell}} f\|_2$ to be small. Even though a function may have a large nominal dimension, d, its effective dimension, i.e., the number of variables that substantially contribute to the variation, must be relatively few, so that the variation under coordinate weights is modest. Under these conditions, the corresponding discrepancy may retain the nearly $O(n^{-1})$ decay.

5.4 Discrepancies for Lattices and Digital Nets

Many useful constructions of LD sequences are constructed numerically by minimizing the discrepancy, however, as noted earlier, computing the discrepancy requires $O(dn^2)$ operations, which can be expensive. However, this computational cost can be reduced for lattices and digital nets when using friendly kernels.

The mean squared discrepancy of a randomly shifted node set, $\{\boldsymbol{x}_i^{\boldsymbol{\Delta}} = \boldsymbol{x}_i + \boldsymbol{\Delta} \bmod \mathbf{1}\}_{i=0}^{n-1}$ for an arbitrary kernel takes the form

$$\begin{aligned}
&\mathbb{E}\big[\text{discrepancy}^2\big(\{\boldsymbol{x}_i^{\boldsymbol{\Delta}}\}_{i=0}^{n-1}, K\big)\big]\\
&\quad= \mathbb{E}\big[\text{discrepancy}^2\big(\{\boldsymbol{x}_i + \boldsymbol{\Delta} \bmod \mathbf{1}\}_{i=0}^{n-1}, K\big)\big]\\
&\quad= \int_{[0,1)^d\times[0,1)^d} K(\boldsymbol{t},\boldsymbol{x})\,\mathrm{d}\boldsymbol{t}\,\mathrm{d}\boldsymbol{x} - \frac{2}{n}\sum_{i=0}^{n-1}\int_{[0,1)^d}\mathbb{E}\big[K(\boldsymbol{t},\boldsymbol{x}_i + \boldsymbol{\Delta} \bmod \mathbf{1})\big]\,\mathrm{d}\boldsymbol{t}\\
&\qquad+ \frac{1}{n^2}\sum_{i,j=0}^{n-1}\mathbb{E}\big[K(\boldsymbol{x}_i + \boldsymbol{\Delta} \bmod \mathbf{1}, \boldsymbol{x}_j + \boldsymbol{\Delta} \bmod \mathbf{1})\big]\\
&\quad= \frac{1}{n^2}\sum_{i,j=0}^{n-1}\mathbb{E}\big[K(\boldsymbol{x}_i - \boldsymbol{x}_j + \boldsymbol{\Delta} \bmod \mathbf{1}, \boldsymbol{\Delta} \bmod \mathbf{1})\big] - \int_{[0,1)^d\times[0,1)^d} K(\boldsymbol{t},\boldsymbol{x})\,\mathrm{d}\boldsymbol{t}\,\mathrm{d}\boldsymbol{x},
\end{aligned}$$

where we have applied a variable transformation $\boldsymbol{\Delta} \to \boldsymbol{\Delta} - \boldsymbol{x}_j$,

$$= \frac{1}{n^2} \sum_{i,j=0}^{n-1} \widetilde{K}(\boldsymbol{x}_i - \boldsymbol{x}_j \bmod \mathbf{1}) - \int_{[0,1)^d} \widetilde{K}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}, \tag{36}$$

and where the filtered version of the reproducing kernel is defined as

$$\widetilde{K}(\boldsymbol{x}) := \int_{[0,1)^d} K(\boldsymbol{x} + \boldsymbol{t} \bmod \mathbf{1}, \boldsymbol{t}) \, \mathrm{d}\boldsymbol{t}. \tag{37}$$

Note that

$$K^{\mathrm{shift}}(\boldsymbol{t}, \boldsymbol{x}) := \widetilde{K}(\boldsymbol{t} - \boldsymbol{x} \bmod \mathbf{1}) \tag{38}$$

is a reproducing kernel. We call it a shift-invariant kernel because $K^{\mathrm{shift}}(\boldsymbol{t} + \boldsymbol{\Delta} \bmod \mathbf{1}, \boldsymbol{x} + \boldsymbol{\Delta} \bmod 1) = \widetilde{K}(\boldsymbol{t} - \boldsymbol{x} \bmod \mathbf{1}) = K^{\mathrm{shift}}(\boldsymbol{t}, \boldsymbol{x})$ for all $\boldsymbol{\Delta} \in [0, 1)^d$.

For arbitrary node sets, this mean squared discrepancy is of similar computational cost as the original, but if the unshifted node set $\{\boldsymbol{x}_i\}_{i=0}^{n-1}$ is a lattice, $\{\boldsymbol{x}_i^{\mathrm{lat}}\}_{i=0}^{n-1}$, as defined in (8), then for any j, the set $\{\boldsymbol{x}_i^{\mathrm{lat}} - \boldsymbol{x}_j^{\mathrm{lat}} \bmod \mathbf{1}\}_{i=0}^{n-1}$ has the same values for all $j = 0, \ldots, n-1$. Thus the mean squared discrepancy for a randomly shifted lattice can be computed in only $O(dn)$ operations, assuming that $K(\boldsymbol{x})$ requires d operations to compute:

$$\begin{aligned} \mathbb{E}\big[\mathrm{discrepancy}^2\big(\{\boldsymbol{x}_i^{\boldsymbol{\Delta}-\mathrm{lat}}\}_{i=0}^{n-1}, K\big)\big] &= \frac{1}{n} \sum_{i=0}^{n-1} \widetilde{K}(\boldsymbol{x}_i) - \int_{[0,1)^d} \widetilde{K}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \\ &= \mathrm{discrepancy}^2\big(\{\boldsymbol{x}_i^{\mathrm{lat}}\}_{i=0}^{n-1}, K^{\mathrm{shift}}\big). \end{aligned} \tag{39}$$

For the kernel defining the centered discrepancy with coordinate weights,

$$\widetilde{K}(\boldsymbol{x}) = \int_{[0,1)^d} \prod_{\ell=1}^{d} \Big[1 + \gamma_\ell^2 \{1/4 - x_\ell(1 - x_\ell)\}\Big], \quad \int_{[0,1)^d} \widetilde{K}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \prod_{\ell=1}^{d} \Big[1 + \frac{\gamma_\ell^2}{12}\Big]. \tag{40}$$

Under modest smoothness assumptions, the discrepancy decays nearly as fast as $O(n^{-1})$ for typical LD sequences. However, when the reproducing kernel, K, is constructed to ensure that functions in $\mathcal{H}_K$ have mixed partial derivatives of up to order r in each coordinate and are periodic in all lower order derivatives, then the discrepancy of lattices can achieve a nearly $O(n^{-r})$ decay rate.

For digital sequences, there is an analogous development of filtering the original kernel by digitally shifts and/or scrambling. This provides a root mean square discrepancy that is again requires only $O(dn)$ operations to compute, not $O(dn^2)$ operations.

5.5 *Randomized Error and Bayesian Error*

The error analysis leading to error bound (25) is a worst-case error analysis, meaning that the error in approximating $\mu(f)$ by $\hat{\mu}_n(f)$ is definitely no larger than the discrepancy times the variation, provided that f lies inside the Hilbert space with reproducing kernel K. We briefly mention other error measures here. More detail is given in [34].

For sample means where the nodes, $\{\boldsymbol{x}_i\}_{i=0}^{n-1}$ are randomized, one can define a measure of their quality based on the root mean squared error of $\hat{\mu}_n(f)$:

$$\mathrm{RMSE}(\hat{\mu}_n(f)) := \sqrt{\mathbb{E}[(\mu(f) - \hat{\mu}_n(f))^2]} \tag{41}$$

$$\leq \underbrace{\sup_{\mathrm{variation}(f,K)\leq 1} \sqrt{\mathbb{E}[(\mu(f) - \hat{\mu}_n(f))^2]}}_{\mathrm{RERR}\left(\{\boldsymbol{x}_i\}_{i=0}^{n-1}, K\right)} \mathrm{variation}(f, K). \tag{42}$$

This quality measure $\mathrm{RERR}\left(\{\boldsymbol{x}_i\}_{i=0}^{n-1}, K\right)$ is typically more optimistic (smaller), than the root mean squared of discrepancy$\left(\{\boldsymbol{x}_i\}_{i=0}^{n-1}, K\right)$ because the latter finds a worst case function for each random instance of nodes, while the former finds just one worst function.

Unfortunately, $\mathrm{RERR}\left(\{\boldsymbol{x}_i\}_{i=0}^{n-1}, K\right)$ is typically not as easy to compute as the discrepancy. If $\hat{\mu}_n$ is unbiased, then $\sqrt{\mathbb{E}[(\mu(f) - \hat{\mu}_n(f))^2]} = \mathrm{std}(\hat{\mu}_n(f))$. For IID nodes we know that this is $\mathrm{std}(f(\boldsymbol{X}))/\sqrt{n}$, which has the same order of decay as root mean squared discrepancy of an IID node set in (35), but a smaller constant.

For randomly shifted lattices $\mathrm{RERR}\left(\{\boldsymbol{x}_i\}_{i=0}^{n-1}, K\right)$ tends to have the same rate of decay as does the root mean squared discrepancy. However, for randomly scrambled and shifted digital nets, there can be an improvement in rate of decay assuming a bit more smoothness [24, 37, 73].

The symmetric, positive definite reproducing kernel, K, which appears in the worst-case error analysis in Sect. 5.1 may instead take the role of the covariance kernel of a Gaussian process with zero mean. Average-case error analysis [77] and probabilistic numerics [6] pursue this approach.

6 Stopping Criteria

When approximating the true μ with $\hat{\mu}_n$, stopping criterion determine how large n should be so that the error $|\mu - \hat{\mu}_n|$ is below an error tolerance ε either certainly or with sufficiently low uncertainty α. The error tolerance and probability threshold are problem dependent. For example, in option pricing one may set the error tolerance to a penny, $\varepsilon = 0.01$, and wish to find error bounds which hold with 95% confidence, $\alpha = 0.05$. The stopping criteria presented here are implemented in `qmcpy` [10].

For simple Monte Carlo with IID nodes, the Central Limit Theorem (CLT) may be used to get an approximate confidence interval $[\mu^-, \mu^+]$ where $\mu^\pm = \hat{\mu} \pm Z^{1-\alpha/2}\sigma/\sqrt{n}, \sigma = \mathrm{Std}[f(\boldsymbol{X})]$ and $Z^{1-\alpha/2}$ is the $1-\alpha/2$ quantile of the standard Gaussian distribution. If

$$n > (2Z^{1-\alpha/2}\sigma/\varepsilon)^2 \tag{43}$$

then $\mu \in (\mu^-, \mu^+]$ with probability $1-\alpha$. A simple IID stopping criterion uses an initial sample of n_0 nodes to approximate $\sigma^2 \approx S^2 := (n_0-1)^{-1}\sum_{i=0}^{n_0-1}(f(\boldsymbol{x}_i) - \hat{\mu})^2$, and then plugs this into (43) to determine n. (Sometimes the sample standard deviation, S, is multiplied by an inflation factor to account for the fact that it is an unbiased estimate and not an upper bound.)

As the CLT holds only as n goes to infinity, this is an approximate stopping criterion. For integrands f with bounded kurtosis, a guaranteed version of this two-stage-sampling idea has been developed to account for the finite sample size using Berry-Esseen inequalities [32].

A straightforward qMC stopping criterion may also be derived from the CLT using several randomizations of a LD node set. Let $\boldsymbol{x}_{ir}$ be the i^{th} node in the r^{th} randomization of a LD node set with $0 \le i < n$ and $1 \le r \le R$. For example, we may use R independent shifts of a lattice node set or independent digital shifts of a digital net. Crucially, the randomized node sets presented in Sect. 4 are also LD. Let $\hat{\mu}^{(r)} = n^{-1}\sum_{i=0}^{n-1} f(\boldsymbol{x}_{ir})$ be sample means for each randomization and set $\hat{\mu}_{nR} = R^{-1}\sum_{r=1}^{R}\hat{\mu}^{(r)}$ to be the aggregate approximation. In the spirit of CLT, a standard approximate $1-\alpha$ confidence interval (μ^-, μ^+) has $\mu^\pm = \hat{\mu}_{nR} \pm t_{R-1}^{1-\alpha/2} S/\sqrt{R}$ where $S^2 = (R-1)^{-1}\sum_{r=1}^{R}(\hat{\mu}^{(r)} - \hat{\mu}_{nR})^2$ and $t_{R-1}^{1-\alpha/2}$ is the $1-\alpha/2$ quantile of the Student t distribution with $R-1$ degrees of freedom. Here we have used the Student t distribution since R is typically small, for example $R = 15$. To use this confidence interval for a stopping criterion, one may iteratively double the sample size n until the interval width is sufficiently low. In [63], comprehensive numerical experiments show that this straightforward qMC stopping criterion achieve better coverage than more involved bootstrap t methods across a variety of test functions.

One of the drawbacks of this qMC-CLT stopping criterion is that, at least intuitively, independent randomizations of LD nodes are less efficient than simply using more LD nodes from a single sequence. For instance, we would expect an approximation with $n = 2^8$ and $R = 2^3$ to perform worse than simply taking $n = 2^{11}$ nodes from a single randomized LD sequence ($R = 1$), but we need R large enough to believe the CLT stopping criterion.

One idea for constructing confidence intervals without replications is to approximate the decay rate of Fourier coefficients of f. Bounds of absolute certainty have be derived for cones of functions with reasonable decay of their Fourier coefficients [34]. As in the qMC-CLT stopping criterion, one may iteratively double n until the bounds are sufficiently tight. Figure 14 compares these decay-tracking qMC algorithms against IID Monte Carlo stopping criterion across a range of tolerances for pricing an Asian option from finance [22, p. 8].

For lattices one tracks the decay of the complex-exponential Fourier coefficients [43]. For digital sequences one tracks the decay of Fourier-Walsh coefficients [33]. In both cases, the definitions of the node sets paired with the appropriate Fourier basis allows these approximate Fourier coefficients to be computed from the data, $\{y_i = f(\boldsymbol{x}_i)\}_{i=0}^{n-1}$ at a cost of $O(n \log n)$ for n a power of 2 using fast transforms.

Another idea which avoids multiple randomizations is to treat f as a Gaussian process, as mentioned in Sect. 5.5. Conditioned on observations $f(\boldsymbol{x}_i)$, a closed form expression exists for the $1-\alpha$ credible intervals for μ. While also applicable when using IID nodes, using LD sequences paired with special kernels enables these credible intervals to be computed at only $O(n \log n)$ cost [40–42]. Again one may iteratively double the sample size n until the credible intervals are sufficiently tight.

For this Bayesian approach, one must find the eigenvalues and eigenvectors of the Gram matrix $\mathsf{K} := \big(K(\boldsymbol{x}_i, \boldsymbol{x}_j)\big)_{i,j=0}^{n-1}$. In general this requires $O(n^3)$ operations. However, pairing shifted lattices with shift-invariant kernels as defined in (38) yields circulant Gram matrices for which computations can be done quickly using the fast Fourier transform [41]. Similarly, pairing digitally shifted digital nets with digitally-shift-invariant kernels yields recursive-block Toeplitz Gram matrices for which computations can be done quickly using the fast Walsh-Hadamard transform [42].

Extensions of these stopping criteria to functions of multiple expectations was explored in [85]. Such problems include computing the posterior mean in a Bayesian context or computing sensitivity indices which quantifying feature importance to variability in a simulation. This work also considered sharing samples across approximations and generalized error criteria, such as a relative error tolerance.

7 Reformulating Our Problem

There are two themes of this section. One is that our original problem may not look like $\mu = f(\boldsymbol{X})$ for $\boldsymbol{X} \sim \mathcal{U}[0,1)^d$, and thus it may need to be rewritten. The second is that qMC does best when most of the variation of f is concentrated in the lower numbered coordinates, and, if possible, we should design f so that this is so.

The methods described in this section for increasing the efficiency of qMC have their parallels in variance reduction methods for IID MC. The difference is that rather than decreasing the variance of $f(\boldsymbol{X})$, we aim to decrease variation(f, K), which appears in error bound (25). Sometimes the same mathematical constructs work for both IID MC and qMC, but not always.

7.1 Variable Transformations

Often qMC problems may be written as $\mu = \mathbb{E}[g(\boldsymbol{T})]$ where $\boldsymbol{T}$ is a d-dimensional random variable with joint density λ. Since qMC nodes have LD with the standard

uniform distribution $\boldsymbol{X} \sim \mathcal{U}[0,1]^d$, we perform an invertible variable transformation $\boldsymbol{\Psi} : [0,1]^d \to \mathcal{M}$ satisfying range$(\boldsymbol{T}) \subseteq \mathcal{M}$. This implies $\mu = \mathbb{E}[f(\boldsymbol{X})]$ where

$$f(\boldsymbol{X}) = g(\boldsymbol{\Psi}(\boldsymbol{X}))\frac{\lambda(\boldsymbol{\Psi}(\boldsymbol{X}))}{\nu(\boldsymbol{\Psi}(\boldsymbol{X}))},$$

and ν is the density of $\boldsymbol{\Psi}(\boldsymbol{X})$. Specifically, $\nu(\boldsymbol{\Psi}(\boldsymbol{X})) = |\boldsymbol{J}_{\boldsymbol{\Psi}}(\boldsymbol{X})|^{-1}$ where $|\boldsymbol{J}_{\boldsymbol{\Psi}}(\boldsymbol{X})|$ is the determinant of the Jacobian of $\boldsymbol{\Psi}$.

The above formulation is equivalent to importance sampling by a distribution with density ν. Optimally, one would choose $\nu \propto g\lambda$ so that f is constant, but this is typically infeasible in practice due to our limited knowledge of g. Choosing $\boldsymbol{\Psi}$ so that $\boldsymbol{T} = \boldsymbol{\Psi}(\boldsymbol{X})$ gives the simplification $f(\boldsymbol{X}) = g(\boldsymbol{\Psi}(\boldsymbol{X}))$. An example of this was given in Sect. 2 where $\boldsymbol{T} \sim \mathcal{N}(\boldsymbol{0}, \mathrm{I}/2)$ and $\boldsymbol{\Psi}(\boldsymbol{X}) = \Phi^{-1}(\boldsymbol{X})/\sqrt{2}$ where Φ^{-1} is the inverse CDF of the standard Gaussian distribution applied elementwise.

As observed in Sect. 5.3, qMC is more effective if the variation of f is concentrated in the lower indexed coordinates. For finance examples where the outcome depends more on the gross behavior of the underlying stochastic process, rather than the high frequency behavior, it makes sense to choose the transformation accordingly. For example, in Fig. 14 we see that if $\boldsymbol{T}$ contains observations from a Brownian motion, then a principal component analysis coupled with the inverse Gaussian distribution shows greater convergence gains compared to the Cholesky decomposition [1].

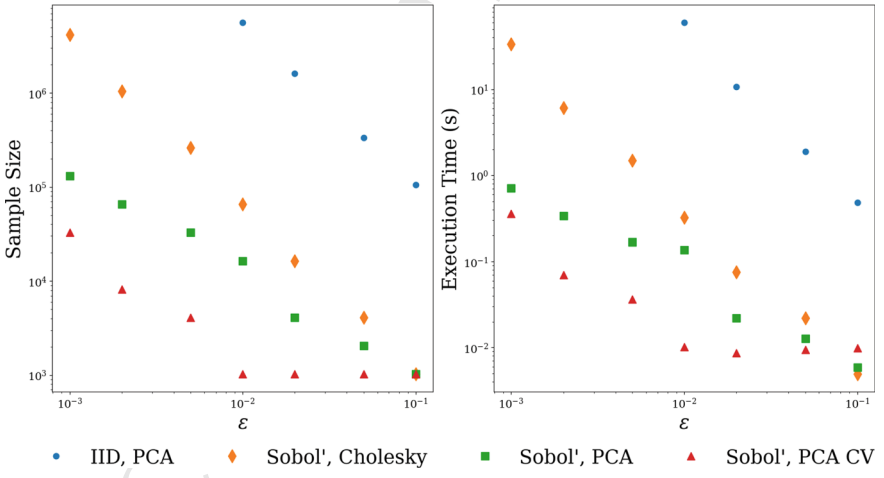


Fig. 14 Comparison of IID Monte Carlo and qMC stopping criteria for estimating the fair price of an Asian option with arithmetic mean. For qMC with Sobol' points, using a principal component analysis (PCA) decomposition of the Brownian motion provides substantial savings compared to the Cholesky decomposition. Using an Asian option with geometric mean control variate (CV) provides additional savings

7.2 Control Variates

Since the upper bound of the qMC error includes the variation of f as a factor, one may improve the efficiency of qMC if one can reformulate the problem to reduce this variation. Choosing a good variable transformation, as mentioned in the previous subsection, is one approach.

Another approach is control variates. If $f_{\text{ctrl}} : [0, 1)^d \to \mathbb{R}$ is a function whose integral is zero, then $\mu(f) = \mu(f - f_{\text{ctrl}})$, and one may approximate $\mu(f)$ by $\hat{\mu}_n(f - f_{\text{ctrl}})$. If the variation of $f - f_{\text{ctrl}}$ is smaller than that of f, then it is likely that $\hat{\mu}_n(f - f_{\text{ctrl}})$ will have smaller error than $\hat{\mu}_n(f)$.

An example of this in quantitative finance is where one may use the payoff of a similar option, whose expectation is known, as a control variate for more exotic options. As noted in [35], reducing the variation of f via control variates may or may not resemble reducing the variance of f. Control variates have been implemented in the stopping criteria based on Fourier coefficients described in the previous section [34]. In Fig. 14 we show that, for pricing an Asian option with arithmetic mean, using an Asian option with geometric mean control variate can provide improved convergence.

7.3 Multilevel Methods

The cost of computing the sample mean defined in (3) is typically $O(dn)$ as the dimension, d, and the sample size, n, tend to infinity. Some applications have a d that is either huge or infinite. One example is in quantitative finance, where d could denote the number of times that the asset price is monitored over the life of a financial derivative times the number of assets involved.

However, the $O(dn)$ cost of approximating μ may often be reduced by employing multilevel methods. In fact, these methods may reduce the cost to the level comparable to solving a low-dimensional approximation [21]. We introduce these methods here.

Let $\{Y_l := f_l(\boldsymbol{X}^{(l)})\}_{l=1}^{L}$ be a sequence of approximations to $Y = f(\boldsymbol{X})$, where $\boldsymbol{X}^{(l)} = (X_1, \ldots, X_l) \sim \mathcal{U}[0, 1)^{d_l}$. Here we assume that $d_1 < \cdots < d_L$, and that the cost of evaluating $f_l(\boldsymbol{x}^{(l)})$ is $O(d_l)$, where $\boldsymbol{x}^{(l)} = (x_1, \ldots, x_l)$. Also, $d_L = d, Y_{d_L} = Y$, and $f_L = f$. Then the population mean may written as a telescoping sum:

$$\begin{aligned}\mu := \mathbb{E}(Y) = \mathbb{E}\big[f(\boldsymbol{X})\big] &= \mathbb{E}(Y_1) + \mathbb{E}(Y_2 - Y_1) + \cdots + \mathbb{E}(Y_{d_L} - Y_{d_{L-1}}) \\ &= \int_{[0,1)^{d_1}} f_1(\boldsymbol{x}^{(1)})\,\mathrm{d}\boldsymbol{x}^{(1)} + \int_{[0,1)^{d_2}} \big[f_2(\boldsymbol{x}^{(2)}) - f_1(\boldsymbol{x}^{(1)})\big]\,\mathrm{d}\boldsymbol{x}^{(2)} + \cdots \\ &\quad + \int_{[0,1)^{d_L}} \big[f_L(\boldsymbol{x}^{(L)}) - f_{L-1}(\boldsymbol{x}^{(L-1)})\big]\,\mathrm{d}\boldsymbol{x}^{(L)}.\end{aligned}$$

This series of integrals may be approximated by a series of sample means, namely

$$\hat{\mu}^{\mathrm{ML}}_{\boldsymbol{n},\boldsymbol{d}} := \frac{1}{n_1}\sum_{i=0}^{n_1-1} f_1(\boldsymbol{x}_i^{(1)}) + \frac{1}{n_2}\sum_{i=0}^{n_2-1}\bigl[f_2(\boldsymbol{x}_i^{(2)}) - f_1(\boldsymbol{x}_i^{(1)})\bigr] + \cdots + \frac{1}{n_L}\sum_{i=0}^{n_L-1}\bigl[f_L(\boldsymbol{x}_i^{(L)}) - f_{L-1}(\boldsymbol{x}_i^{(L-1)})\bigr], \quad (44)$$

where $\boldsymbol{n} = (n_1, \ldots, n_L)$ and $\boldsymbol{d} = (d_1, \ldots, d_L)$. If C_1 is the computational cost of $f^{(1)}(\boldsymbol{x}^{(1)})$ and C_l is the computational cost of $f^{(l)}(\boldsymbol{x}^{(1)}) - f_{l-1}(\boldsymbol{x}_i^{(l-1)})$ for $l = 2, \ldots, L$, then the total computational cost of (44) is

$$\mathrm{cost}(\boldsymbol{n}, \boldsymbol{d}) = n_1 C_1 + \cdots + n_L C_L. \quad (45)$$

A worst case error analysis as described in Sect. 5 implies that

$$\bigl|\mu - \hat{\mu}^{\mathrm{ML}}_{\boldsymbol{n},\boldsymbol{d}}\bigr| \le V_1/n_1 + V_2/n_2 + \cdots + V_L/n_L, \quad (46)$$

where the V_l are defined such that we expect them to have little dependence on n_l since we are using quasi-Monte Carlo nodes:

$$V_1 = n_1 \times \mathrm{discrepancy}\bigl(\{\boldsymbol{x}_i^{(1)}\}_{i=0}^{n_2-1}, K^{(1)}\bigr) \times \mathrm{variation}(f^{(1)}, K^{(1)})$$
$$V_l = n_l \times \mathrm{discrepancy}\bigl(\{\boldsymbol{x}_i^{(l)}\}_{i=0}^{n_l-1}, K^{(l)}\bigr) \times \mathrm{variation}(f^{(l)} - f^{(l-1)}, K^{(l)}), \quad l = 2, \ldots, L,$$

and $K^{(l)}$ is the reproducing kernel corresponding to the relevant Hilbert space of functions of d_l variables.

Optimizing vector of sample sizes $\boldsymbol{n}$ to minimize the error bound in (46) for a fixed cost and assuming constant $\boldsymbol{C} = (C_1, \ldots, C_L)$ and $\boldsymbol{V} = (V_1, \ldots, V_L)$, yields $\bigl|\mu - \hat{\mu}^{\mathrm{ML}}_{\boldsymbol{n},\boldsymbol{d}}\bigr| \le \varepsilon$ for

$$n_l = \frac{\sqrt{V_l}\bigl(\sqrt{V_1 C_1} + \cdots + \sqrt{V_L C_L}\bigr)}{\varepsilon\sqrt{C_l}}, \qquad l = 1, \ldots, d,$$
$$\mathrm{cost}(\boldsymbol{n}, \boldsymbol{d}) = \frac{\bigl(\sqrt{V_1 C_1} + \cdots + \sqrt{V_L C_L}\bigr)^2}{\varepsilon}. \quad (47)$$

We see that the cost of approximating μ within an absolute error tolerance of ε is $O(\varepsilon^{-1})$, as expected, but the dependence of $\mathrm{cost}(\boldsymbol{n}, \boldsymbol{d})$ on the dimension of the problem depends critically on how quickly the variation at each level decays. Suppose that

– The cost of a function value at each level, l, is proportional to the dimension, d_l, and that the dimensions *increase* exponentially, i.e., $d_l = d_1^l$ and $C_l \le \alpha d_1^l$.
– The V_l, which are roughly the variation, *decrease* exponentially, i.e., $V_l \le \beta r^{-l}$.

Then the computational cost in (47) can be bounded above by

$$\begin{aligned}\mathrm{cost}(\boldsymbol{n}, \boldsymbol{d}) &\le \frac{\alpha\beta\bigl((d_1/r)^{1/2} + \ldots + (d_1/r)^{L/2}\bigr)^2}{\varepsilon} \\ &= \frac{\alpha\beta\bigl((d_1/r)^{1/2} - (d_1/r)^{(L+1)/2}\bigr)^2}{\varepsilon\bigl(1 - (d_1/r)^{1/2}\bigr)^2}.\end{aligned}$$

If the V_l decay quickly enough, which means that the corrections $f_l - f_{l-1}$ depending on many variables decay quickly enough, in particular, if $r > d_1$, then

$$\mathrm{cost}(\boldsymbol{n}, \boldsymbol{d}) \le \frac{\alpha\beta d_1}{r\varepsilon\bigl(1 - (d_1/r)^{1/2}\bigr)^2} = O(d_1/\varepsilon). \tag{48}$$

In this case $\mathrm{cost}(\boldsymbol{n}, \boldsymbol{d})$ for this high dimensional problem is not much worse than the cost of ensuring that just error of the rough d_1 approximation is smaller than ε, which would be $V_1 C_1/\varepsilon \le \alpha\beta d_1/(r\varepsilon)$.

However, if $r < d_1$, then $\mathrm{cost}(\boldsymbol{n}, \boldsymbol{d}) \ge V_L C_L/\varepsilon$ which may be as large as $\alpha\beta(d_1/r)^{-L}\varepsilon$, which grows exponentially as $d = d_L = d_1^L$ increases via increasing L. Granted, this may not be as high a cost as the standard single-level approach where the cost would be roughly $V_1 C_L$ if the V_l are decaying.

It is an art to write one's problem to have the favorable case of high or even infinite nominal dimension, d, be solvable with a computational cost comparable to the low-dimensional case. However, the multilevel method takes advantage of the principle noted in Sect. 5.3, namely that qMC thrives when problem have a low effective dimension, even if the nominal dimension is large.

8 Beyond Computing Expectations

Expectations of the form $\mu := \mathbb{E}(f(\mathbf{X}))$ appear in several prominent areas of mathematics and qMC methods have long been applied in these contexts. Namely, in mathematical finance many problems can be formulated as high-dimensional integrals, where the large number of dimensions arises from small time steps in time discretization and/or a large number of variables. In this setting, Monte Carlo methods are applied to randomly simulate the evolution of stock prices over time, and the expected value is then estimated by averaging over all sample paths. The qMC approach replaces the randomly generated paths by carefully constructed paths in an effort to reduce the computational burden on the required number of paths. Excellent survey works for qMC and randomized qMC applied to finance include [7, 39, 57, 61, 90].

Another fruitful domain for qMC methods is in partial differential equations (PDEs), used for computing the expectations of nonlinear functionals of solutions of certain classes of PDEs. This includes problems in areas like fluid dynamics or material science, where random coefficients in the PDE model uncertain properties; see [23, 55, 56] and references therein for qMC applied to PDEs.

QMC methods have also found recent applications in scientific machine learning of PDE solutions. It has been shown that LD nodes are more efficient than IID nodes for training neural networks modeling PDE solution [62, 66]. LD nodes have also been used to more efficiently fit Gaussian process regression models for operator learning the solution process of PDEs with uncertain coefficients [84]. There the computational speedup follows from pairing LD nodes with special kernels to induce nicely structured gram matrices, as used in the fast Bayesian cubature stopping criteria discussed in Sect. 6.

Besides computing expectations, qMC methods have found many applications in other areas of the computational sciences. We lightly review these applications to highlight their wider practical impact and to motivate the further development of qMC theory and methodology.

For example, when considering a variable Y, it is often useful to determine its distribution, F, or the density, ϱ, i.e.,

$$F(y) := \mathbb{P}(Y \le y) = \mathbb{P}(f(\mathbf{X}) \le y) = \int_{[0,1]^d} \mathbf{1}_{(-\infty, y]}(f(\mathbf{x}))d\mathbf{x}, \quad \varrho(y) := F'(y).$$

Here, $F(y)$ gives the cumulative probability that Y does not exceed y, while $\varrho(y)$ represents the corresponding density function. Several recent works for density estimation use qMC [20, 59, 60] alongside randomized qMC [4] and often incorporate kernel-based methods or rely on conditioning strategies (sometimes called preintegration).

Keller [47] was the first to adopt qMC for rendering photorealistic images in computer graphics where the problem of simulating light transport can be mathematically represented as an integral over all possible light paths from source to camera. Many algorithms involving sampling with LD nodes have since been developed much further in [47–50, 52, 79]. (Remarkably, MC and qMC methods have even been awarded with an Academy Award for visual effects in movie making [71, 88].)

Path planning is a critical aspect of efficient robotic operation, defining of a route from an initial position to a target destination while considering an optimality criterion. Most existing approaches utilize LD nodes, such as the Halton sequence [89, 93] with more recent efforts [8] demonstrating that employing state-of-the-art LD nodes from [78] can significantly enhance the efficiency and flexibility of a robotic arm, demonstrated in both toy and real-world tasks.

In summary, qMC methods have demonstrated their applicability and impact well beyond computing expectations, finding fruitful applications across fields such as computer graphics, robotics, and machine learning.

9 Conclusion

Quasi-Monte Carlo methods, based on low-discrepancy sequences, can approximate the mean of a function, f, of d variables to within an error tolerance of ε at a cost of roughly $O(\varepsilon^{-1})$ function evaluations, provided that the problem is formulated so that f depends primarily on a modest number of variables. This means that qMC outperforms alternative methods such as simple Monte Carlo and sampling on grids. QMC is particularly powerful when the effective dimension of the problem is small, even though the nominal dimension, d, may be large. Software libraries are available to facilitate qMC computations.

Acknowledgements This work is supported by National Science Foundation DMS Grant No. 2316011. The third author would like to acknowledge support from the DOE Office of Science Graduate Student Research (SCGSR) Program.

References

1. Acworth, P., Broadie, M., Glasserman, P.: A comparison of some Monte Carlo techniques for option pricing. In: Niederreiter, H., Hellekalek, P., Larcher, G., Zinterhof, P. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 1996. Lecture Notes in Statistics, vol. 127, pp. 1–18. Springer-Verlag, New York (1998)
2. Aronszajn, N.: Theory of reproducing kernels. Trans. Am. Math. Soc. **68**, 337–404 (1950)
3. Atanassov, E.I.: On the discrepancy of the Halton sequences. Math. Balkanica, New Series **18**(D1–2), 15–32 (2004)
4. Ben Abdellah, A., L'Ecuyer, P., Owen, A.B., Puchhammer, F.: Density estimation by randomized quasi-Monte Carlo. SIAM/ASA J. Uncertain. Quantif. **9**, 280–301 (2021). https://doi.org/10.1137/19M1259213
5. Berlinet, A., Thomas-Agnan, C.: Reproducing Kernel Hilbert Spaces in Probability and Statistics. Kluwer Academic Publishers, Boston (2004)
6. Briol, F.X., Oates, C.J., Girolami, M., Osborne, M.A., Sejdinovic, D.: Probabilistic integration: a role in statistical computation? Stat. Sci. **34**, 1–22 (2019)
7. Caflisch, R.E., Morokoff, W.: Valuation of mortgage backed securities using the quasi-Monte Carlo method. In: International Association of Financial Engineers First Annual Computational Finance Conference (1996)
8. Chahine, M., Rusch, T.K., Patterson, Z.J., Rus, D.: Improving efficiency of sampling-based motion planning via Message-Passing Monte Carlo (2024). arXiv preprint arXiv:2410.03909
9. Chi, H., Mascagni, M., Warnock, T.: On the optimal Halton sequence. Math. Comput. Simul. **70**(1), 9–21 (2005)
10. Choi, S.C.T., Hickernell, F.J., Jagadeeswaran, R., McCourt, M., Sorokin, A.: QMCPy: A Quasi-Monte Carlo Python Library (versions 1–1.6) (2025). https://doi.org/10.5281/zenodo.3964489, https://qmcsoftware.github.io/QMCSoftware/
11. Clément, F.: An optimization perspective on the construction of low-discrepancy point sets. Ph.D. thesis, Sorbonne Université (2024)
12. Clément, F., Doerr, C., Klamroth, K., Paquete, L.: Constructing optimal l_∞ star discrepancy sets (2024). arXiv preprint arXiv:2311.17463
13. Clément, F., Doerr, C., Paquete, L.: Star discrepancy subset selection: problem formulation and efficient approaches for low dimensions. J. Complex. **70**, 101,645 (2022)

14. Clément, F., Doerr, C., Paquete, L.: Heuristic approaches to obtain low-discrepancy point sets via subset selection. J. Complex. **83**, 101,852 (2024)
15. Darmon, Y., Godin, M., L'ecuyer, P., Jemel, A., Marion, P., Munger, D.: LatNet builder (2018). https://github.com/umontreal-simul/latnetbuilder
16. Dick, J., Kritzer, P., Pillichshammer, F.: Lattice Rules: Numerical Integration, Approximation, and Discrepancy. Springer Series in Computational Mathematics. Springer Cham (2022). https://doi.org/10.1007/978-3-031-09951-9
17. Dick, J., Kuo, F.Y.: Reducing the construction cost of the component-by-component construction of good lattice rules. Math. Comp. **73**, 1967–1988 (2004)
18. Dick, J., Pillichshammer, F.: Digital Nets and Sequences: Discrepancy Theory and Quasi-Monte Carlo Integration. Cambridge University Press, Cambridge (2010)
19. Faure, H., Lemieux, C.: Generalized Halton sequences in 2008: a comparative study **19**(4) (2009)
20. Gilbert, A.D., Kuo, F.Y., Sloan, I.H.: Analysis of preintegration followed by quasi-Monte Carlo integration for distribution functions and densities. SIAM J. Numer. Anal. **61**, 135–166 (2023)
21. Giles, M.B.: Multilevel Monte Carlo methods. Acta Numer. **24**, 259–328 (2015). https://doi.org/10.1017/S096249291500001X
22. Glasserman, P.: Monte Carlo Methods in Financial Engineering, Applications of Mathematics, vol. 53. Springer-Verlag, New York (2004)
23. Graham, I.G., Kuo, F.Y., Nuyens, D., Scheichl, R., Sloan, I.H.: Quasi-Monte Carlo methods for elliptic PDEs with random coefficients and applications. J. Comput. Phys. **230**, 3668–3694 (2011). https://doi.org/10.1016/j.jcp.2011.01.023
24. Heinrich, S., Hickernell, F.J., Yue, R.X.: Optimal quadrature for Haar wavelet spaces. Math. Comp. **73**, 259–277 (2004)
25. Hickernell, F.J., Kirk, N., Sorokin, A.G.: (2025). https://github.com/QMCSoftware/QMCSoftware/blob/MCQMC2024/demos/talk_paper_demos/MCQMC2024Tutorial/MCQMC2024Tutorial.ipynb
26. Hickernell, F.J.: A comparison of random and quasirandom points for multidimensional quadrature. In: Niederreiter, H., Shiue, P.J.S. (eds.) Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing. Lecture Notes in Statistics, vol. 106, pp. 213–227. Springer-Verlag, New York (1995)
27. Hickernell, F.J.: A generalized discrepancy and quadrature error bound. Math. Comp. **67**, 299–322 (1998). https://doi.org/10.1090/S0025-5718-98-00894-1
28. Hickernell, F.J.: Lattice rules: How well do they measure up? In: Hellekalek, P., Larcher, G. (eds.) Random and Quasi-Random Point Sets. Lecture Notes in Statistics, vol. 138, pp. 109–166. Springer-Verlag, New York (1998)
29. Hickernell, F.J.: What affects the accuracy of quasi-Monte Carlo quadrature? In: Niederreiter, H., Spanier, J. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 1998, pp. 16–55. Springer-Verlag, Berlin (2000)
30. Hickernell, F.J., Hong, H.S.: Quasi-Monte Carlo methods and their randomizations. In: Chan, R., Kwok, Y.K., Yao, D., Zhang, Q. (eds.) Applied Probability, AMS/IP Studies in Advanced Mathematics, vol. 26, pp. 59–77. American Mathematical Society, Providence, Rhode Island (2002)
31. Hickernell, F.J., Hong, H.S., L'Écuyer, P., Lemieux, C.: Extensible lattice sequences for quasi-Monte Carlo quadrature. SIAM J. Sci. Comput. **22**, 1117–1138 (2000). https://doi.org/10.1137/S1064827599356638
32. Hickernell, F.J., Jiang, L., Liu, Y., Owen, A.B.: Guaranteed conservative fixed width confidence intervals via Monte Carlo sampling. In: Dick, J., Kuo, F.Y., Peters, G.W., Sloan, I.H. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2012. Springer Proceedings in Mathematics and Statistics, vol. 65, pp. 105–128. Springer-Verlag, Berlin (2013). https://doi.org/10.1007/978-3-642-41095-6
33. Hickernell, F.J., Jiménez Rugama, Ll.A.: Reliable adaptive cubature using digital sequences. In: Cools, R., Nuyens, D. (eds.) Monte Carlo and Quasi-Monte Carlo Methods: MCQMC, Leuven, Belgium, April 2014. Springer Proceedings in Mathematics and Statistics, vol. 163, pp. 367–383. Springer-Verlag, Berlin (2016). arXiv:1410.8615 [math.NA]

610417_1_En_1_Chapter-online ☑ TYPESET ☐ DISK ☐ LE ☑ CP Disp.:13/12/2025 Pages: 37 Layout: T1-Standard

34. Hickernell, F.J., Jiménez Rugama, Ll.A., Li, D.: Adaptive quasi-Monte Carlo methods for cubature. In: Dick, J., Kuo, F.Y., Woźniakowski, H. (eds.) Contemporary Computational Mathematics—a Celebration of the 80th Birthday of Ian Sloan, pp. 597–619. Springer-Verlag (2018). https://doi.org/10.1007/978-3-319-72456-0
35. Hickernell, F.J., Lemieux, C., Owen, A.B.: Control variates for quasi-Monte Carlo. Stat. Sci. **20**, 1–31 (2005). https://doi.org/10.1214/088342304000000468
36. Hickernell, F.J., Niederreiter, H.: The existence of good extensible rank-1 lattices. J. Complex. **19**, 286–300 (2003)
37. Hickernell, F.J., Yue, R.X.: The mean square discrepancy of scrambled (t, s)-sequences. SIAM J. Numer. Anal. **38**, 1089–1112 (2000). https://doi.org/10.1137/S0036142999358019
38. Hinrichs, A., Oettershagen, J.: Optimal point sets for quasi-monte carlo integration of bivariate periodic functions with bounded mixed derivatives. In: Monte Carlo and Quasi-Monte Carlo Methods 2016, pp. 385–405 (2018)
39. Jäckel, P.: Monte Carlo Methods in Finance. John Wiley & Sons Ltd., Chichester, England (2002)
40. Jagadeeswaran, R.: Fast automatic Bayesian cubature using matching kernels and designs. Ph.D. thesis, Illinois Institute of Technology (2019)
41. Jagadeeswaran, R., Hickernell, F.J.: Fast automatic Bayesian cubature using lattice sampling. Stat. Comput. **29**, 1215–1229 (2019). https://doi.org/10.1007/s11222-019-09895-9
42. Jagadeeswaran, R., Hickernell, F.J.: Fast automatic Bayesian cubature using Sobol' sampling. In: Botev, Z., Keller, A., Lemieux, C., Tuffin, B. (eds.) Advances in Modeling and Simulation: Festschrift in Honour of Pierre L'ecuyer, pp. 301–318. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-10193-9_15
43. Jiménez Rugama, Ll.A., Hickernell, F.J.: Adaptive multidimensional integration based on rank-1 lattices. In: Cools, R., Nuyens, D. (eds.) Monte Carlo and Quasi-Monte Carlo Methods: MCQMC, Leuven, Belgium, April 2014. Springer Proceedings in Mathematics and Statistics, vol. 163, pp. 407–422. Springer-Verlag, Berlin (2016). arXiv:1411.1966
44. Joe, S., Kuo, F.Y.: Remark on algorithm 659: implementing Sobol's quasirandom sequence generator. ACM Trans. Math. Softw. **29**, 49–57 (2003)
45. Joe, S., Kuo, F.Y.: Constructing Sobol sequences with better two-dimensional projections. SIAM J. Sci. Comput. **30**, 2635–2654 (2008)
46. Keister, B.D.: Multidimensional quadrature algorithms. Comput. Phys. **10**, 119–122 (1996). https://doi.org/10.1063/1.168565
47. Keller, A.: A quasi-Monte Carlo algorithm for the global illumination problem in a radiosity setting. In: Niederreiter, H., Shiue, P.J. (eds.) Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, pp. 239–251. Springer-Verlag, New York (1995)
48. Keller, A.: Quasi-Monte Carlo radiosity. In: Pueyo, X., Schröder, P. (eds.) Eurographics Rendering Workshop, pp. 101–110 (1996)
49. Keller, A.: Quasi-Monte Carlo image synthesis in a nutshell. In: Dick, J., Kuo, F.Y., Peters, G.W., Sloan, I.H. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2012. Springer Proceedings in Mathematics and Statistics, vol. 65, pp. 213–249. Springer-Verlag, Berlin (2013). https://doi.org/10.1007/978-3-642-41095-6
50. Keller, A., Premoze, S., Raab, M.: Advanced (quasi) Monte Carlo methods for image synthesis. In: ACM SIGGRAPH 2012 Courses, SIGGRAPH '12. Association for Computing Machinery, New York, NY, USA (2012)
51. Kirk, N., Lemieux, C.: An improved Halton sequence for implementation in quasi-Monte Carlo methods. In: Proceedings of the Winter Simulation Conference, pp. 431–442. IEEE Press, Piscataway, NJ (2024)
52. Kollig, T., Keller, A.: Efficient bidirectional path tracing by randomized quasi-Monte Carlo integration. In: Fang, K.T., Hickernell, F.J., Niederreiter, H. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2000, pp. 290–305. Springer-Verlag, Berlin (2002)
53. Korobov, N.M.: Number-theoretic methods in approximate analysis. Goz. Izdat. Fiz.-Math. (1963)

54. Kuo, F.Y.: Component-by-component constructions achieve the optimal rate of convergence for multivariate integration in weighted Korobov and Sobolev spaces. J. Complex. **19**, 301–320 (2003)
55. Kuo, F.Y., Nuyens, D.: Application of quasi-Monte Carlo methods to elliptic PDEs with random diffusion coefficients—a survey of analysis and implementation. Found. Comput. Math. **16**, 1631–1696 (2016). https://people.cs.kuleuven.be/~dirk.nuyens/qmc4pde/
56. Kuo, F.Y., Schwab, C., Sloan, I.H.: Quasi-Monte Carlo finite element methods for a class of elliptic partial differential equations with random coefficients. SIAM J. Numer. Anal. **50**, 3351–3374 (2012). https://doi.org/10.1137/110845537
57. L'Ecuyer, P.: Quasi-Monte Carlo methods with applications in finance. Financ. Stoch. **13**, 307–349 (2009)
58. L'Ecuyer, P., Marion, P., Godin, M., Puchhammer, F.: A tool for custom construction of QMC and RQMC point sets. In: Keller, A. (ed.) Monte Carlo and Quasi-Monte Carlo Methods 2020, pp. 51–70. Springer, Cham (2022)
59. L'Ecuyer, P., Puchhammer, F.: Density estimation by Monte Carlo and quasi-Monte Carlo. In: Keller, A. (ed.) Monte Carlo and Quasi-Monte Carlo Methods 2020, pp. 3–21. Springer, Cham (2022)
60. L'Ecuyer, P., Puchhammer, F., Ben Abdellah, A.: Monte Carlo and quasi–Monte Carlo density estimation via conditioning. Inf. J. Comput. **34**(3), 1729–1748 (2022). https://doi.org/10.1287/ijoc.2021.1135
61. Lemieux, C.: Randomized quasi-Monte Carlo: a tool for improving the efficiency of simulations in finance. In: Ingalls, R.G., Rossetti, M.D., Smith, J.S., Peters, B.A. (eds.) Proceedings of the 2004 Winter Simulation Conference, pp. 1565–1573. IEEE Press (2004)
62. Longo, M., Mishra, S., Rusch, T.K., Schwab, C.: Higher-order quasi-Monte Carlo training of deep neural networks. SIAM J. Sci. Comput. **43**(6), A3938–A3966 (2021)
63. L'Ecuyer, P., Nakayama, M.K., Owen, A.B., Tuffin, B.: Confidence intervals for randomized quasi-Monte Carlo estimators. In: 2023 Winter Simulation Conference (WSC), pp. 445–456. IEEE (2023)
64. Maize, E.: Contributions to the theory of error reduction in quasi-Monte Carlo methods. Ph.D. thesis, The Claremont Graduate School (1981)
65. Matoušek, J.: On the L_2-discrepancy for anchored boxes. J. Complex. **14**, 527–556 (1998)
66. Mishra, S., Rusch, T.K.: Enhancing accuracy of deep learning algorithms by training with low-discrepancy sequences. SIAM J. Numer. Anal. **59**(3), 1811–1834 (2021)
67. Niederreiter, H.: Random number generation and Quasi-Monte Carlo methods. IN: CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, Philadelphia (1992)
68. Novak, E., Woźniakowski, H.: Tractability of Multivariate Problems Volume II: Standard Information for Functionals, No. 12 in EMS Tracts in Mathematics. European Mathematical Society, Zürich (2010)
69. Nuyens, D., Cools, R.: Fast algorithms for component-by-component construction of rank-1 lattice rules in shift-invariant reproducing kernel Hilbert spaces. Math. Comp. **75**, 903–920 (2006)
70. Nuyens, D., Cools, R.: Fast component-by-component construction. In: Niederreiter, H., Talay, D. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2004, pp. 373–387. Springer-Verlag, Berlin (2006)
71. Oscars: The 86th scientific & technical awards 2013|2014 (2014). https://www.oscars.org/sci-tech/ceremonies/2014
72. Owen, A.B.: Randomly permuted (t, m, s)-nets and (t, s)-sequences. In: Niederreiter, H., Shiue, P.J.S. (eds.) Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing. Lecture Notes in Statistics, vol. 106., pp. 299–317. Springer-Verlag, New York (1995)
73. Owen, A.B.: Scrambled net variance for integrals of smooth functions. Ann. Stat. **25**, 1541–1562 (1997)
74. Owen, A.B., Pan, Z.: Gain coefficients for scrambled Halton points. SIAM J. Numer. Anal. **62**(3), 1021–1038 (2024)

610417_1_En_1_Chapter-online ☑ TYPESET ☐ DISK ☐ LE ☑ CP Disp.:13/12/2025 Pages: 37 Layout: T1-Standard

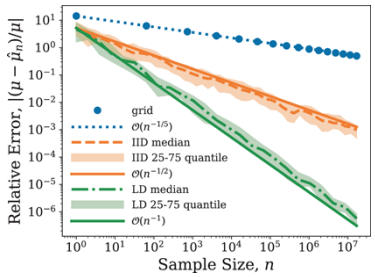
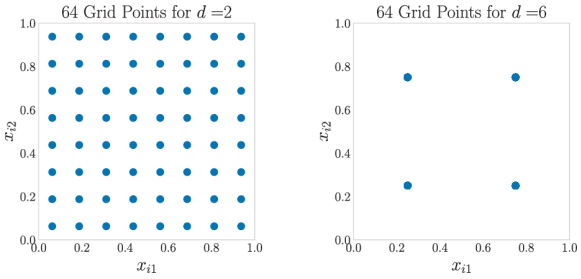
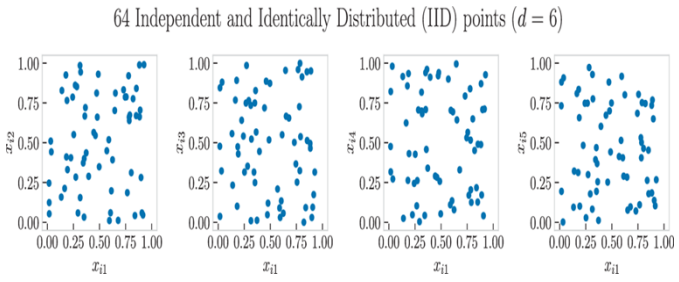
75. Pan, Z., Owen, A.B.: Super-polynomial accuracy of one dimensional randomized nets using the median of means. Math. Comp. **92**, 805–837 (2023). https://doi.org/10.1090/mcom/3791
76. Paulin, L., Bonneel, N., Coeurjolly, D., Iehl, J.C., Keller, A., Ostromoukhov, V.: Matbuilder: mastering sampling uniformity over projections. ACM Trans. Graph. (Proceedings of SIGGRAPH) **41**(4) (2022). DOI https://doi.org/10.1145/3528223.3530063
77. Ritter, K.: Average-Case Analysis of Numerical Problems. Lecture Notes in Mathematics, vol. 1733. Springer-Verlag, Berlin (2000)
78. Rusch, T.K., Kirk, N., Bronstein, M.M., Lemieux, C., Rus, D.: Message-passing Monte Carlo: generating low-discrepancy point sets via graph neural networks. Proc. Natl. Acad. Sci. **121**(40), e2409913,121 (2024)
79. Shirley, P., Edwards, D., Boulos, S.: Monte Carlo and quasi-Monte Carlo methods for computer graphics. In: Keller, A., Heinrich, S., Niederreiter, H. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2006, pp. 167–177. Springer-Verlag, Berlin (2008)
80. Sloan, I.H.: Qmc integration—beating intractability by weighting the coordinate directions. In: Fang, K.T., Hickernell, F.J., Niederreiter, H. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2000, pp. 103–123. Springer-Verlag, Berlin (2002)
81. Sloan, I.H., Joe, S.: Lattice Methods for Multiple Integration. Oxford University Press, Oxford (1994)
82. Sloan, I.H., Woźniakowski, H.: When are quasi-Monte Carlo algorithms efficient for high dimensional integrals? J. Complex. **14**, 1–33 (1998)
83. Sobol', I.M.: The distribution of points in a cube and the approximate evaluation of integrals. U.S.S.R. Comput. Math. Math. Phys. **7**, 86–112 (1967)
84. Sorokin, A.G., Pachalieva, A., O'Malley, D., Hyman, J.M., Hickernell, F.J., Hengartner, N.W.: Computationally efficient and error aware surrogate construction for numerical solutions of subsurface flow through porous media. Adv. Water Resour. **193**, 104,836 (2024)
85. Sorokin, A.G., Rathinavel, J.: On bounding and approximating functions of multiple expectations using quasi-Monte Carlo. In: International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, pp. 583–599. Springer (2022)
86. Tuffin, B.: A new permutation choice in Halton sequences. In: Hellekalek, P., Larcher, G., Niederreiter, H., Zinterhof, P. (eds.) Proceedings of the Conference on Monte Carlo and Quasi-Monte Carlo Methods. Lecture Notes in Statistics, vol. 127. Springer, Berlin, New York, pp. 427–435 (1998)
87. Vandewoestyne, B., Cools, R.: Good permutations for deterministic scrambled Halton sequences in terms of L2-discrepancy. J. Comput. Appl. Math. **189**, 341–361 (2006). Proceedings of the 11th International Congress on Computational and Applied Mathematics
88. Veach, E.: Robust Monte Carlo methods for light transport simulation. Ph.D. thesis, Stanford University (1997)
89. Velagić, J., Delimustafić, D., Osmanković, D.: Mobile robot navigation system based on probabilistic road map (PRM) with Halton sampling of configuration space. In: IEEE 23rd International Symposium on Industrial Electronics (ISIE), pp. 1227–1232 (2014)
90. Wang, X., Sloan, I.H.: Why are high-dimensional finance problems often of low effective dimension? SIAM J. Sci. Comput. **27**(1), 159–183 (2005)
91. Weyl, H.: Über die Gleichverteilung der Zahlen mod Eins. Math. Ann. **77**, 313–352 (1916)
92. Winker, P., Fang, K.T.: Optimal U-type designs. In: Niederreiter, H., Hellekalek, P., Larcher, G., Zinterhof, P. (eds.) Monte Carlo and quasi-Monte Carlo methods 1996. Lecture Notes in Statistics, vol. 127, pp. 436–448. Springer-Verlag, New York (1998)
93. Zhong, H., Cong, M., Wang, M., Du, Y., Liu, D.: HB-RRT: a path planning algorithm for mobile robots using Halton sequence-based rapidly-exploring random tree. Eng. Appl. Artif. Intell. **133**, 108,362 (2024)

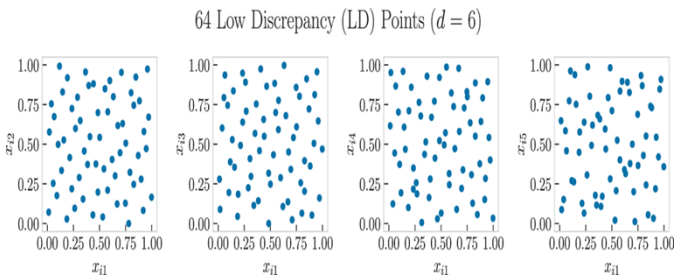
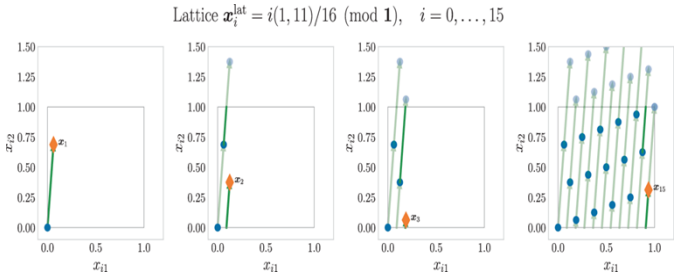
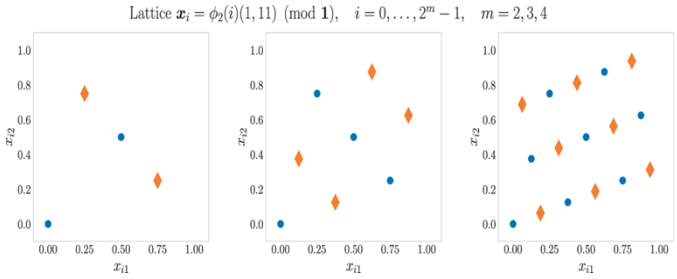
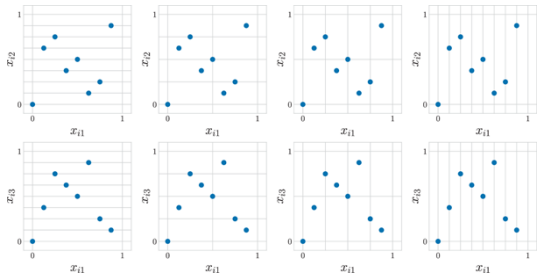
Author Queries

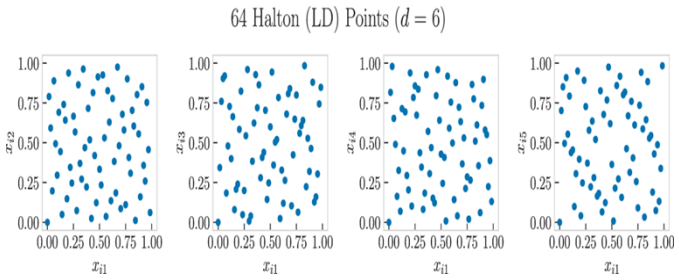
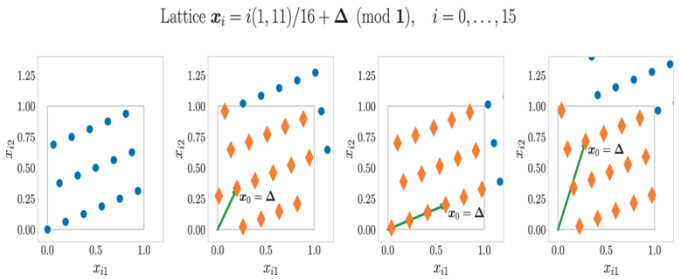
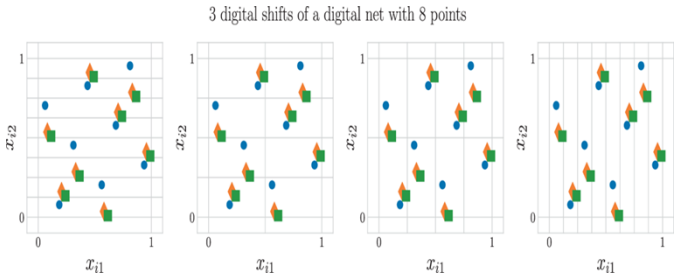
Chapter 1

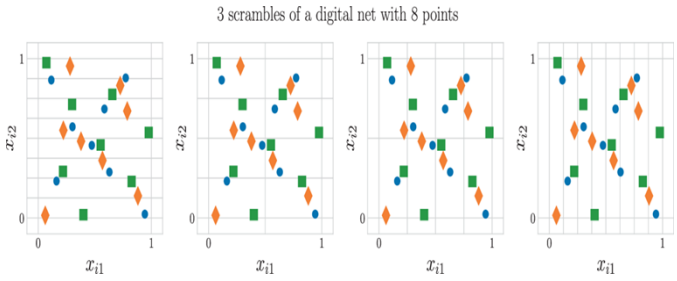
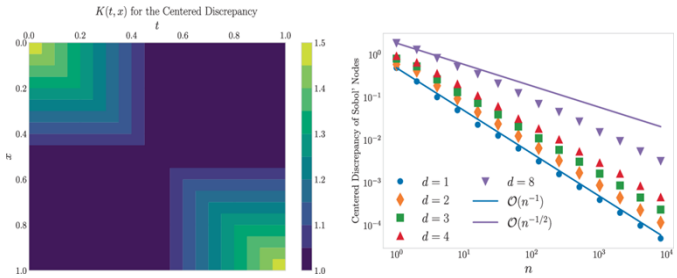
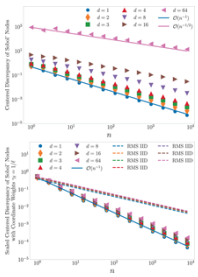
Query Refs.	Details Required	Author's response
AQ1	Kindly check and verify that the author name(s) and the identification of the corresponding author(s) are correctly recognized and presented in the correct sequence order and spellings, i.e., given name, middle name/initial, and family name for the authors. In addition, please verify that the E-mail addresses and Affiliation(s) of the corresponding author(s) and co-author(s) shown on the metadata page are valid, and make any necessary amendments if required.	
AQ2	Please check and confirm the inserted citation of Fig. 11 is correct. If not, please suggest an alternate citation. Please note that figure should be cited in sequential order in the text.	

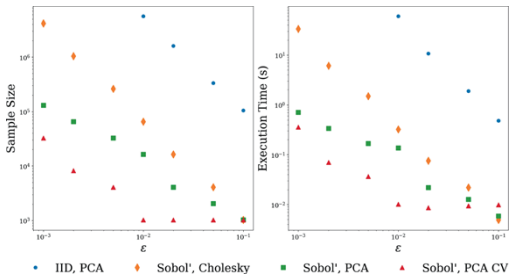
Alternative Texts for Your Images, Please Check and Correct them if Required

Page no	Fig/Photo	Thumbnail	Alt-text Description
4	Fig1		Chart showing relative error versus sample size on a logarithmic scale. The y-axis represents relative error, and the x-axis represents sample size, . Three main trends are depicted: a blue dotted line for grid data with a rate of mathcal{O}(^{-1/5}) , an orange dashed line for IID data with a rate of mathcal{O}(^{-1/2}) , and a green dashed-dotted line for LD data with a rate of mathcal{O}(^{-1}) . Shaded areas indicate the 25-75 quantile ranges for IID and LD data.
5	Fig2		Two X-Y charts comparing grid points for different dimensions. The left chart shows 64 grid points for dimension d=2, with points evenly distributed across the plane. The right chart displays 4 grid points for dimension d=5, with points more sparsely distributed. Both charts have axes labeled x_1 and x_2, ranging from 0.0 to 1.0.
6	Fig3		Scatter plots showing 64 independent and identically distributed (IID) points across four panels. Each panel represents a different pair of variables: x_1 vs x_2, x_1 vs x_3, x_1 vs x_4, and x_1 vs x_5. The axes range from 0.00 to 1.00, with points scattered throughout each plot. The title indicates the dimension d = 6.

Page no	Fig/Photo	Thumbnail	Alt-text Description
6	Fig4		Scatter plots showing 64 low discrepancy points for dimensions d = 6. Each plot displays points distributed along the x and y axes, labeled x_1 and x_2, respectively. The points are spread across the range from 0.00 to 1.00 on both axes, illustrating the distribution of data in a low discrepancy sequence.
8	Fig5		Four-panel figure showing X-Y charts illustrating lattice transformations. Each panel displays a grid with points and lines, demonstrating changes in the distribution of points over iterations. The x and y axes are labeled x_1 and x_2, respectively, ranging from 0 to 1. The title indicates a lattice transformation with the formula x^t = (t, 11)/16 mod 1 for t = 0, ldots, 15. Points are marked in blue and red, with lines connecting them, showing progression across panels.
8	Fig6		Three-panel X-Y chart showing lattice points. Each panel displays a scatter plot with blue and orange markers. The x-axis is labeled as x_1 and the y-axis as x_2 . The title indicates a lattice sequence with parameters phi(0;1,1) and m = 2, 3, 4 . The distribution of points varies across panels, illustrating different configurations.
11	Fig7		A grid of eight scatter plots, each showing data points plotted on an X-Y axis. The x-axis is labeled x_{i1} and the y-axis is labeled x_{i2} for the top row and x_{i3} for the bottom row. Each plot displays a different distribution pattern of blue dots, indicating variations in data relationships across the

Page no	Fig/Photo	Thumbnail	Alt-text Description
			plots.
11	Fig8		Scatter plots showing 64 Halton low-discrepancy points across four panels. Each panel plots points with x_1 on the x-axis and x_2, x_3, x_4, and x_5 on the y-axes respectively. The title indicates the dimension d = 6. Each plot displays a uniform distribution of points within a unit square.
14	Fig9		Four-panel X-Y chart illustrating lattice points with the formula x_i = i(1,11)/16 + Delta , (mod , 1), , i = 0, ldots, 15 . Each panel shows a different arrangement of blue circles and orange diamond shapes on a grid. The second and fourth panels include a green line and the label x_0 = Delta . The x-axis is labeled x_1 and the y-axis is labeled x_2 .
15	Fig10		Four X-Y charts illustrating digital shifts of a digital net with 8 points. Each chart displays points represented by different shapes: circles, squares, and triangles, distributed across the grid. The x-axis is labeled X_{i1} and the y-axis is labeled X_{i2}, both ranging from 0 to 1. The title above the charts reads "3 digital shifts of a digital net with 8 points."

Page no	Fig/Photo	Thumbnail	Alt-text Description
16	Fig11		Four X-Y charts displaying scrambled data points with different colors and shapes, including blue circles, green squares, and orange diamonds. Each chart has axes labeled x_1 and x_3, with a title above reading "3 scrambles of a digital net with 8 points." The charts illustrate variations in data distribution across the panels.
21	Fig12		Contour map and line graph figure. The left panel is a contour map with a color gradient from dark blue to green, representing values from 0.0 to 1.5. The x and y axes range from 0.0 to 1.0, labeled as x and y, respectively. The right panel is a log-log plot showing the context discrepancy of several models. The x-axis represents ranging from 10^2 to 10^4, and the y-axis represents the context discrepancy. Different shapes and colors represent dimensions d = 1 to d = 4, with lines indicating theoretical bounds O(^{-1}) and O(^{-1/d}).
23	Fig13		The image consists of two X-Y charts. The top chart shows the "Centered Discrepancy of Sobol' Nodes" against on a logarithmic scale, with data points for different dimensions d (1, 2, 3, 4, 8, 16, 64) represented by various shapes and colors. Two trend lines, O(^{-1}) and O(^{-1/2}) , are included. The bottom chart displays the "Scaled Centered Discrepancy of Sobol' Nodes with Coordinate Weights gamma = 1/ell" against , also on a logarithmic scale, with similar data points and trend lines. Both charts include a legend for dimensions and

Page no	Fig/Photo	Thumbnail	Alt-text Description
			trend lines.
29	Fig14		Two-panel figure showing X-Y charts. The left chart plots sample size against epsilon (ε) on a logarithmic scale, with different markers representing methods: blue circles for IID, PCA; orange diamonds for Sobol', Cholesky; green squares for Sobol', PCA; and red triangles for Sobol', PCA CV. The right chart plots execution time in seconds against epsilon (ε) on a logarithmic scale, using the same markers for methods. Both charts illustrate the relationship between epsilon and the respective metrics for various computational methods.