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

Fred J. Hickernell[1], Nathan Kirk[2], and Aleksei G. Sorokin[3]

[1] Department of Applied Mathematics, Illinois Institute of Technology, Chicago, IL, 60616, USA
hickernell@iit.edu, http://www.iit.edu/ hickernell
[2] nkirk@iit.edu
[3] asorokin@hawk.iit.edu

Abstract. Many problems in quantitative finance, uncertainty quantification, and other disiplines 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 methods. We also briefly describe the use of quasi-Monte Carlo methods for problems beyond computing the mean, μ.

Keywords:

1 Introduction

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

- Future market forces, which affect financial risk,
- The porosity of rock, which affects the extraction of oil or gas, or
- Elemetary 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 multidude 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 a particular input value, $\boldsymbol{x}$, the corresponding outcome $y = f(\boldsymbol{x})$ can

be computed by an algorithm, whose complexity may be such that f can be considered a "black box". The user selects a sequence of input values, also known as data sites or *nodes*, $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots$, which give rise to the observed outcomes $y_0 = f(\boldsymbol{x}_0), y_1 = f(\boldsymbol{x}_1), \ldots$. The y_i form the basis for approximating $\mathbb{E}(Y)$, quartiles of Y, the probability density of Y, and other quantities of interest.

Simple Monte Carlo chooses the nodes $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots\}$ to be independent and identically distributed (IID). Quasi-Monte Carlo (qMC) chooses the nodes *not* IID, but with an empirical distribution that approximates well the true probability distribution of $\boldsymbol{X}$. The difference between these two 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 session illustrates by example of the advantages of qMC. This is followed by a description of how deterministic LD sequences are constructed and randomizations that preserve their low discrepancy. We describe various measures of discrepancy. We explain how to decide what sample size, n, is sufficient to meet the user's error criterion. We then discuss how to rewrite the problem of interest in a qMC-friendly way.

2 An Illustration of Quasi-Monte Carlo

We illustrate the benefits of qMC with an example from Keister [6, (13)], motivated by computational physics:

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

where $\|\boldsymbol{t}\| := \sqrt{t_1^2 + \cdots + t_d^2}$. 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.

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

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

This does require some further preparation, which is discussed in greater generality in Section 7.

This μ, may be thought of as 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., $\boldsymbol{T} := (T_1, \ldots, T_d) \sim \mathcal{N}(\boldsymbol{0}, \mathrm{I}/2)$:

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

Nearly all LD sequences, which underly qMC, are defined to approximate the standard uniform distribution, $\mathcal{U}[0,1]^d$. Thus, we perform a variable transformation $bst = \big(\Phi^{-1}(x_1), \ldots, \Phi^{-1}(x_d)\big)/\sqrt{2}$, where is 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:

$$\begin{aligned}\mu = \int_{[0,1]^d} \underbrace{\pi^{d/2}\cos\Big(\big\|\big(\Phi^{-1}(x_1), \ldots, \Phi^{-1}(x_d)\big)\big\|/\sqrt{2}\Big)}_{=:f(\boldsymbol{x})} \,\mathrm{d}\boldsymbol{x} \\ = \mathbb{E}[Y] = \mathbb{E}[f(\boldsymbol{X})], \qquad \boldsymbol{X} \sim \mathcal{U}[0,1]^d.\end{aligned} \tag{5}$$

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

$$\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^{n} pi^{d/2}\cos\Big(\big\|\big(\Phi^{-1}(x_{i1}), \ldots, \Phi^{-1}(x_{id})\big)\big\|/\sqrt{2}\Big). \tag{6}$$

Consider the specific case of $d = 6$ for which $\mu \approx -2.327303729298$. We approximate μ by the sample mean defined in (3) for three different kinds of nodes, $\{\boldsymbol{x}_i\}_{i=0}^{n-1}$, for various sample sizes, n, and plot the relative errors, $|(\mu - \hat{\mu}_n)/\mu|$ in Figure 1. The three kind of nodes are

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

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%.
- *Grid nodes.* Although grids may be attractive for low dimensional problems, e.g., $d = 1$, 2, or 3, they 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 Figure 1.
  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.

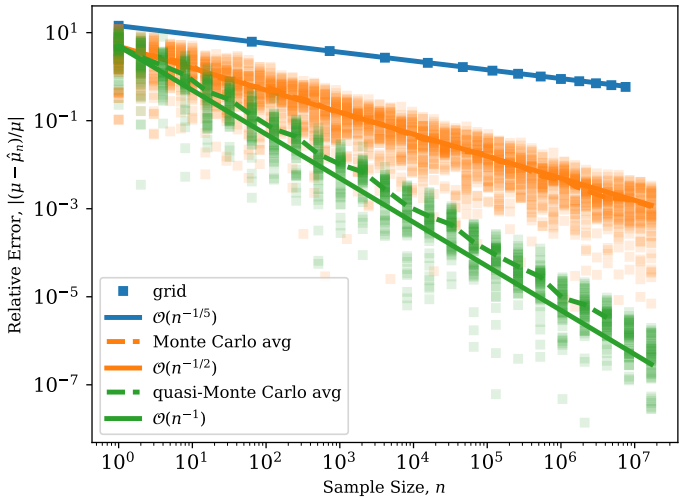

Fig. 1. The relative error of approximating μ defined in (5) by the sample mean, $\hat{\mu}_n$, defined in (3) for various choices of nodes, $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots$. Grids have the largest error and LD nodes have the smallest error.

– *IID nodes.* Simple MC (orange) is a substantial improvement over grid nodes. As a reminder, 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 Figure 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 (Figure 2), all IID points may be seen when viewed in any lower dimensional projection, as seen in Figure 3. The disadvantage of IID points 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 nodes.* For qMC methods the error 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 sizes. For this Sobol' sequence, n is chosen to be non-negative integer powers of 2, which tends to give better accuracy than arbitrary n.
Figure 4 shows typical two-dimensional projections of a 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 Section 5. Constructions of LD node sequences are explained in the next section.

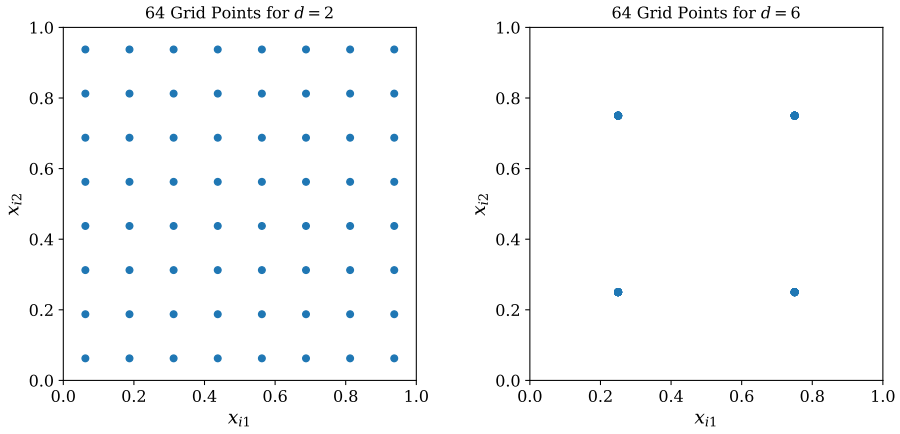

Fig. 2. Although a two-dimensional grid covers the unit square rather well, a d-dimensional grid for modest d does *not* cover the unit cube well. For example, one can only see four distinct points in a two dimensional projection of a six-dimensional grid with 64 points.

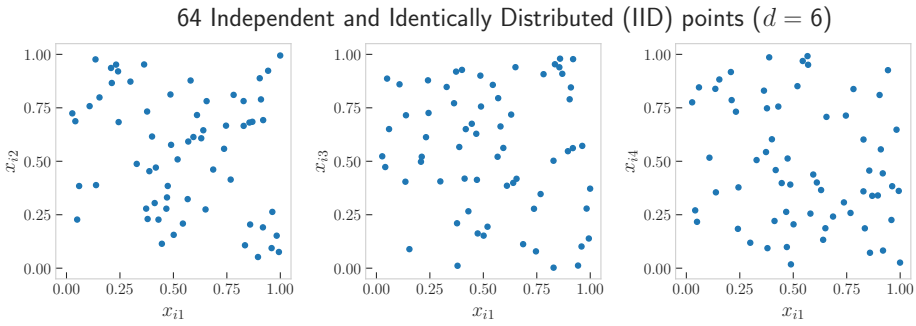

Fig. 3. IID points cover the unit cube better than grid points, although one does observe clusters and gaps. In any projection there is a similar looking distribution of all 64 points

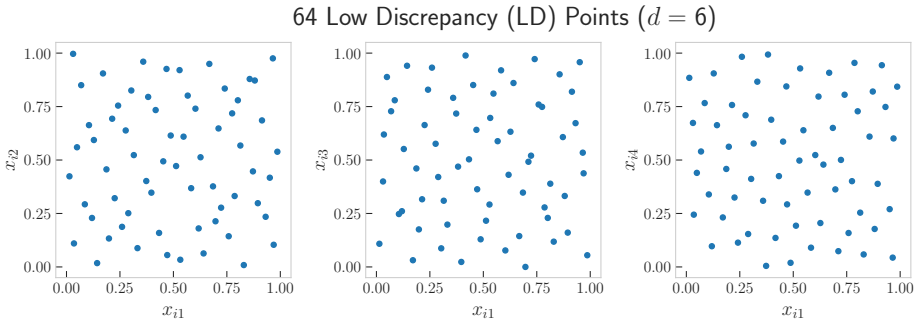

Fig. 4. LD points cover the unit cube better even better than IID points or grids. In any projection there is a similar looking distribution of all 64 points.

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 to one another, whether they be deterministic or randomized.

3.1 Lattice Sequences

One of the simplest LD constructions is the family of good lattice points [2,10]. As a finite node set, they are defined as follows:

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

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 $\{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}\}$ defined in (8) is closed under addition modulo $\mathbf{1}$ so that it forms a group.

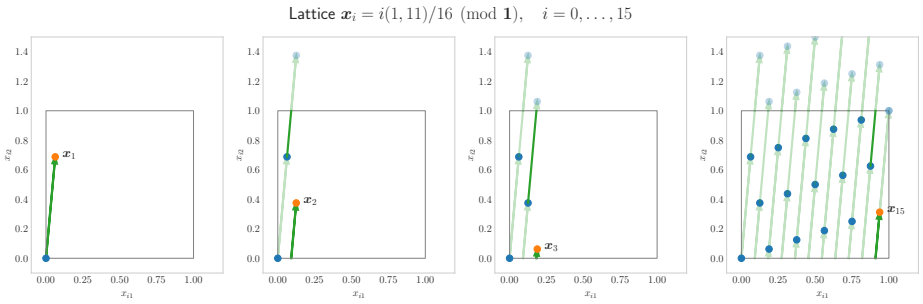

Fig. 5. The construction of a good lattice point set in two dimensions with 16 nodes is obtained by moving $\boldsymbol{h}/n$ beyond the present point and wrapping around the boundaries of the square until one returns to the origin.

One disadvantage of this construction is that it is not readily extensible. For the example in Figure 5 the first 8 points do not fill the unit square well. However, the set of points defined by even i do a reasonable job. This suggests a method for defining extensible lattice sequences that was proposed independently by [8] and [4].

Let's start with the one-dimensional extensible lattice that is defined by the van der Corput sequence, $\{\phi_b(0), \phi_b(1), \ldots\}$. This involves reflecting the digits of the integer i in base b about the decimal point. For example, $\phi_2(6) = \phi_2(110_2) = {}_20.011 = 3/8$. In general,

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

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

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

This reordering of points from the original construction allows us to preserve the lattice structure for the first b^m points for any non-negative integer m. That is, $\{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{2^m-1}\}$ is a closed under addition modulo $\mathbf{1}$.

Figure 6 shows the first 4, 8, and 16 points of the earlier example in Figure 5. The blue nodes are a copy of the nodes in the plot to the left, and the the orange nodes correspond to a shifted copy of the blue nodes modulo $\mathbf{1}$. For the left plot, the shift is $\phi_2(2)\boldsymbol{h} = (1, 11)/4$ or equivalently $(0.25, 0.75)$. For the middle plot, the shift is $\phi_2(4)\boldsymbol{h} = (1, 11)/8$ or equivalently $(0.125, 0.375)$. For the right plot, the shift is $\phi_2(8)\boldsymbol{h} = (1, 11)/16$ or equivalently $(0.0625, 0.6875)$.

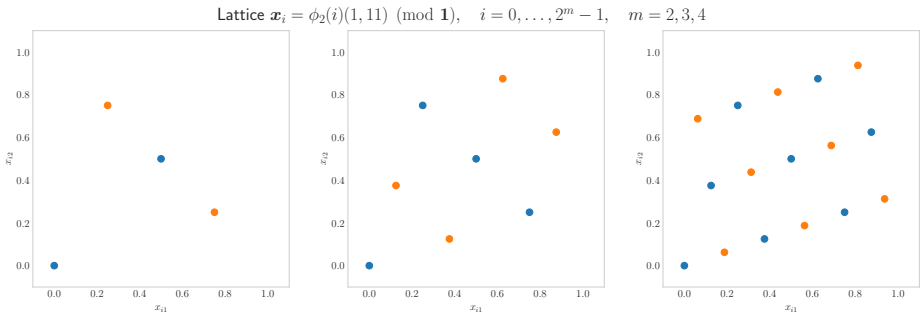

Fig. 6. An extensible lattice corresponding to Figure 5 with the nodes reordered using the van der Corput sequence in base 2. For each plot the blue nodes are a copy of the nodes to the left and the orange nodes are a shifted copy (modulo $\mathbf{1}$) of the blue nodes.

For the example in Figure 6, increasing n beyond 16 repeats the original points 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 [5], where the existence of good generating vectors is also proved. In practice, one searches computationally for a $\boldsymbol{h}$ that produces good set of lattice nodes of size $n = b^m$ for a range of m [4].

The construction of great generating vectors for lattices has attracted a great deal of interest. The component-by-component constructions pioneered by Frances Kuo, Josef Dick, Ian Sloan, Dirk Nuyens and others are surveyed in [2, Chapter ?]. LatNet builder [7,1] is a software package by Pierre L'Ecuyer and collaborators, which constructs generators for lattices and digital sequences.

3.2 Digital Sequences

Another family of LD sequences also built upon the van der Corput sequence (9) is digital sequences [3,9]. 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., $3/8 \oplus 3/4 = {}_2 0.011 \oplus {}_2 0.110 = {}_2 0.101 = 5/8$. A digital sequence is defined as

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

where $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ are carefully chosen. The node set $\{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{2^m-1}\}$ for non-negative integer m is called a digital net and is closed under $\oplus$.

If $x_{ij\ell}$ denotes the ℓ^{th} digit of the j^{th} coordinate of $\boldsymbol{x}_i$, then the literature often defines generating matrices, $\mathsf{C}_j = (x_{ij\ell})_{\ell,i=1}^{M,N}$ for $j = 1, \ldots, d$, where M is th maximum number of bits in the expression for $\boldsymbol{x}_i$, say 52, and 2^N is the maximum number of nodes that is intended to be generated. The digital sequence can then be defined equivalently as

$$\begin{pmatrix} x_{ij1} \\ x_{ij2} \\ \vdots \end{pmatrix} = \mathsf{C}_j \begin{pmatrix} i_0 \\ i_1 \\ \vdots \end{pmatrix} \pmod 2, \qquad j = 1, \ldots, d, \ i = 0, 1, \ldots. \tag{12}$$

To understand how these points might be evenly distributed over $[0,1)^d$, imagine a box of the form

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

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

$$\begin{gathered} \text{the first } k_1 \text{ rows of the first } m \text{ columns of } \mathsf{C}_1 \text{ plus} \\ \text{the first } k_2 \text{ rows of the first } m \text{ columns of } \mathsf{C}_2 \text{ plus} \\ \cdots \\ \text{the first } k_d \text{ rows of the first } m \text{ columns of } \mathsf{C}_d \\ \text{are linearly independent over the finite field with the elements } \{0,1\}, \end{gathered} \tag{14}$$

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 the smallest t for which condition (14) holds for all $\boldsymbol{k}$ with $\|\boldsymbol{k}\|_1 \le m - t$. Equivalently, this t is the smallest value for which 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 of the form (12) for which this condition holds for all non-negative m is called a (t, d)-sequence.

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

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

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 $2^{m-\|\boldsymbol{k}\|_1} = 2$ points. 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 $k_1 = 2$, $k_2 = 0$, and $k_3 = 1$ 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

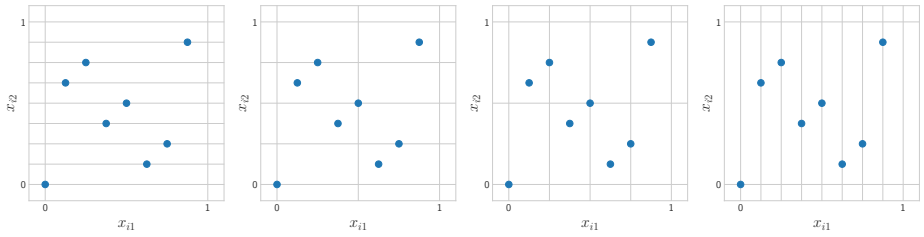

Fig. 7. Each box in the two dimensional projections of the node set (15) plotted above show one point in each box except for the second row, third plot from the left. Thus, this node set is not a $(0, 3, 3)$-net. It is however a $(1, 3, 3)$-net.

Digital sequence generators can be found via number theory or numerical search [**?**]. The earliest instance is due to Sobol' [**?**].

3.3 Halton sequences

While lattice sequences and digital sequences have preferred sample sizes, n, Halton sequences have no preferred sample size. The Halton sequence is defined in terms of the van der Corput sequence

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

where $b_1, \ldots, b_d$ is a choice of d distinct prime numbers. Often they are chosen as the first d prime numbers. Figure 8 shows two dimensional projections of Halton points

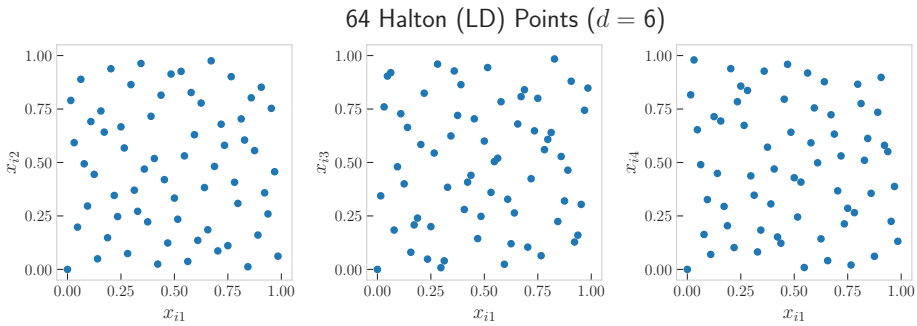

Fig. 8. Halton points as defined in (16) have low discrepancy but no preferred sample size.

3.4 LD Nodes by Optimization

[Nathan Kirk write here]

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 some advantages randomization has advantages as well.

- Randomization done right removes bias in the estimator.
- 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 Section 1, as seen in (6). There 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 $\boldsymbol{x}_0 = \boldsymbol{0}$, as can be seen in (8), (11), (16), and Figures 5–8. The node $\boldsymbol{0}$ 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.

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 $\boldsymbol{1}$ so that shifted extensible lattice turns (10) into

$$\boldsymbol{x}_i^{\boldsymbol{\Delta}\text{-lat}} := \boldsymbol{x}_i + \boldsymbol{\Delta} \pmod{\boldsymbol{1}} = \phi_b(i)\boldsymbol{h} + \boldsymbol{\Delta} \pmod{\boldsymbol{1}}, \qquad i = 0, 1, \ldots. \tag{17}$$

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

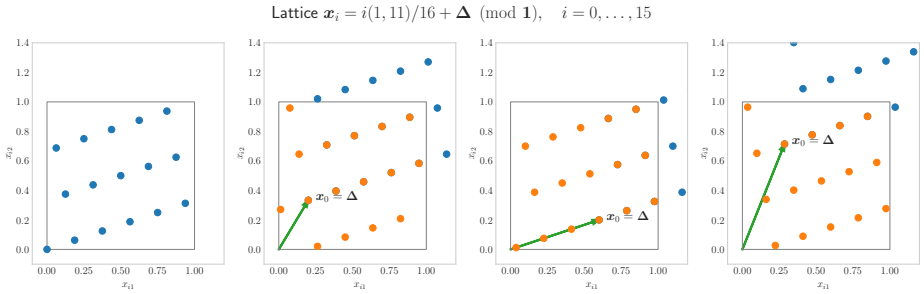

Fig. 9. The original unshifted lattice (left) and three different shifts modulo $\mathbf{1}$ of the lattice. Note that the structure of the lattice is maintained.

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

$$\boldsymbol{x}_i^{\boldsymbol{\Delta}\text{-dig}} := \boldsymbol{x}_i \oplus \boldsymbol{\Delta} = i_0\boldsymbol{x}_1 \oplus i_1\boldsymbol{x}_2 \oplus i_2\boldsymbol{x}_4 \oplus \cdots \oplus \boldsymbol{\Delta} \in [0,1)^d \quad \text{for } i = i_0 + i_1 2 + i_2 2^2 + \cdots . \tag{18}$$

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

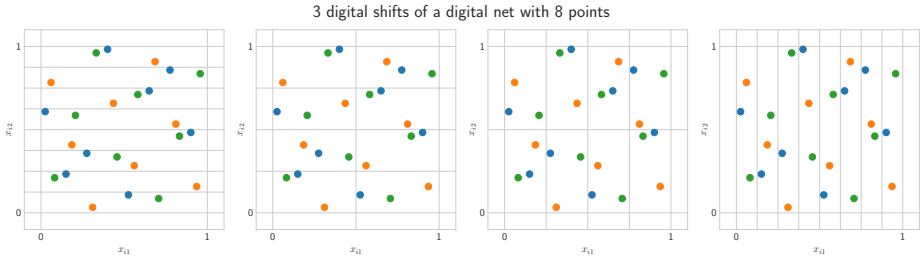

Fig. 10. Three digital shifts of a digital $(0, 3, 2)$-net with nodes denoted by three different colors. There is one node in each smaller box of the form (13) for $\|\boldsymbol{k}\| = 3$, for each digitally shifted net.

Let $\{\boldsymbol{x}_i^{\boldsymbol{\Delta}}\}_{i=0}^{n-1}$ denote a random shift of any deterministic original node set, $\{\boldsymbol{x}_i\}_{i=0}^{n-1}$ as described above. 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 at the beginning of this section.

4.2 Digital Scrambles

For digital nets one can randomize even further by a scrambling that preserves the t-value. Owen [**?**] proposed a scrambling of nets called nested uniform scrambling. A simpler version called linear matrix scrambling was proposed by Matoušek [**?**] 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:

$$\begin{pmatrix} x_{ij1}^{\mathrm{scr}} \\ x_{ij2}^{\mathrm{scr}} \\ \vdots \end{pmatrix} = \mathsf{L_j C}_j \begin{pmatrix} i_0 \\ i_1 \\ \vdots \end{pmatrix} + \begin{pmatrix} \Delta_{1j} \\ \Delta_{2j} \\ \vdots \end{pmatrix} \pmod 2, \qquad j = 1, \ldots, d, \ i = 0, 1, \ldots, \tag{19a}$$

$$\mathsf{L_j} := \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots \\ l_{21} & 1 & 0 & 0 & \cdots \\ l_{31} & l_{32} & 1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \ddots \end{pmatrix}, \qquad l_{\ell j} \overset{\text{IID}}{\sim} \mathcal{U}\{0, 1\} \tag{19b}$$

A random digital shift, $\boldsymbol{\Delta}$ is also added, where $\Delta_{\ell j}$ denotes the ℓ^{th} digit of the j^{th} of the shift.

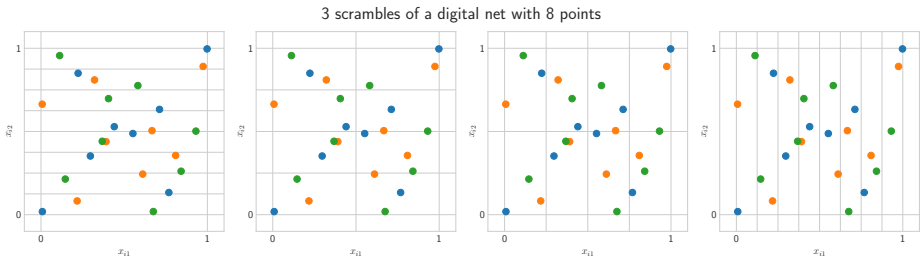

Fig. 11. Three linear scrambles of a digital $(0, 3, 2)$-net with nodes denoted by three different colors. There is one node in each smaller box of the form (13) for $\|\boldsymbol{k}\| = 3$, for each digitally shifted net.

[make a scramble of an image]

4.3 Other Randomizations for Halton

4.4 Randomized Generators

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 way minimizing discrepancy leads to better approximations to μ.

5.1 Discrepancies Defined by Kernels

Let $K : [0,1)^d \times [0,1)^2 \to \mathbb{R}$ be a function satisfying the following two conditions:

$$\text{Symmetry:} \quad K(\boldsymbol{t},\boldsymbol{x}) = K(\boldsymbol{x},\boldsymbol{t}) \qquad \forall \boldsymbol{t},\boldsymbol{x} \in [0,1)^d \tag{20a}$$

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

$$\text{distinct } \boldsymbol{x}_0,\ldots,\boldsymbol{x}_{n-1} \in [0,1)^d. \tag{20b}$$

Then it is known [**?**] 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

$$\text{Belonging:} \quad K(\cdot,\boldsymbol{x}) \in \mathcal{H}_K \qquad \forall \boldsymbol{x} \in [0,1)^d \tag{21a}$$

$$\text{Reproducing:} \quad f(\boldsymbol{x}) = \langle K(\cdot,\boldsymbol{x}), f\rangle \qquad \forall \boldsymbol{x} \in [0,1)^d,\ f \in \mathcal{H}_K. \tag{21b}$$

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 Theorm, and η_L is called the representer of L. Using the reproducing property of K, one may derove am explict 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:

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

If $\mu(\cdot)$ is a bounded linear functional on $\mathcal{H}_K$ is, 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-Schawarz inequality imply a tight error bound:

$$\begin{aligned} |\mu(f) - \hat{\mu}_n(f)| &= |\langle \eta_{\mu(\cdot)-\hat{\mu}_n(\cdot)}, f\rangle_{\mathcal{H}_K}| \le \|\eta_{\mu(\cdot)-\hat{\mu}_n(\cdot)}\|_{\mathcal{H}_K}\, \|f\|_{\mathcal{H}_K} \\ |\mu(f) - \hat{\mu}_n(f)| &\le \underbrace{\|\eta_{\mu(\cdot)-\hat{\mu}_n(\cdot)}\|_{\mathcal{H}_K}}_{\text{discrepancy}\left(\{\boldsymbol{x}\}_{i=0}^{n-1},K\right)} \underbrace{\inf_{c\in\mathbb{R}} \|f - c\|_{\mathcal{H}_K}}_{\text{variation}(f,K)} \qquad \forall f \in \mathcal{H}_K, \end{aligned} \tag{22}$$

since $\hat{\mu}_n$ is exact for constants. (It is assumed that $\mathcal{H}$ 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 as

$$\begin{aligned}\|\eta_{\mu(\cdot)-\hat{\mu}_n(\cdot)}\|_{\mathcal{H}_K}^2 &= \bigl(\mu(\cdot\cdot) - \hat{\mu}_n(\cdot\cdot)\bigr)\Bigl(\bigl(\mu(\cdot) - \hat{\mu}_n(\cdot)\bigr)\bigl(K(\cdot,\cdot\cdot)\bigr)\Bigr)\\ &= \int_{[0,1)^d\times[0,1)^d} K(\boldsymbol{t},\boldsymbol{x})\,\mathrm{d}\boldsymbol{t}\,\mathrm{d}\boldsymbol{x} - \frac{2}{n}\sum_{i=0}^{n-1}\int_{[0,1)^d} K(\boldsymbol{t},\boldsymbol{x}_i)\,\mathrm{d}\boldsymbol{t}\\ &\quad + \frac{1}{n^2}\sum_{i,j=0}^{n-1} K(\boldsymbol{x}_i,\boldsymbol{x}_j). \end{aligned} \tag{23}$$

Error bound (22) has two factors:

- the discrepancy, which is the norm of the error functional, is a measure of deficiency of the node set, $\{\boldsymbol{x}\}_{i=0}^{n-1}$, and
- the variation is a measure of the roughness of the function defining our random variable whose expectation we wish to compute.

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 Section 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 discussed in Section 7.

Although error bound (22) is elegant, it leaves several matters unresolved:

- The reproducing kernel K defines both the discrepancy and the variation. This means that differnt 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 ones assumptions about f, such as smoothness or periodicity. Knowing K, however, does not automatically lead to an explict formula for the inner product of the associated Hilbert space, $\langle\cdot,\cdot\rangle_{\mathcal{H}_K}$. Conversely, having an explict formula for $\langle\cdot,\cdot\rangle_{\mathcal{H}_K}$ does not automatically lead to an explict 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 "black box" f lies in $\mathcal{H}_K$.
- Moreover, even if one is confident that $f \in \mathcal{H}_K$, it is usually impractical to compute variation(f, K), as we shall see below. Error bound (22) cannot be used as a stopping criterion to determine the n needed to satisfy an error tolerance.

Nonetheless (22)

5.2 Geometrically Motivated Discrepancies

5.3 Coordinate weights

5.4 Discrepancies for Lattices and Nets

5.5 Randomized Error

6 Stopping Criteria

Art
Tony
Jags

7 Reformulating Our Problem

7.1 Variable Transformations

7.2 Variation Reduction

7.3 Multilevel Methods

8 Ongoing Research

9 Conclusion

References

1. Darmon, Y., Godin, M., L'Ecuyer, P., Jemel, A., Marion, P., Munger, D.: LatNet builder (2018). URL https://github.com/umontreal-simul/latnetbuilder
2. Dick, J., Kritzer, P., Pillichshammer, F.: Lattice Rules: Numerical Integration, Approximation, and Discrepancy. Springer Series in Computational Mathematics. Springer Cham (2022). DOI https://doi.org/10.1007/978-3-031-09951-9
3. Dick, J., Pillichshammer, F.: Digital Nets and Sequences: Discrepancy Theory and Quasi-Monte Carlo Integration. Cambridge University Press, Cambridge (2010)
4. 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). DOI 10.1137/S1064827599356638
5. Hickernell, F.J., Niederreiter, H.: The existence of good extensible rank-1 lattices. J. Complexity **19**, 286–300 (2003)
6. Keister, B.D.: Multidimensional quadrature algorithms. Computers in Physics **10**, 119–122 (1996). DOI 10.1063/1.168565
7. L'Ecuyer, P., Marion, P., Godin, M., Puchhammer, F.: A tool for custom construction of QMC and RQMC point sets. In: A. Keller (ed.) Monte Carlo and Quasi-Monte Carlo Methods: MCQMC, Oxford, 2020, Springer Proceedings in Mathematics and Statistics. Springer, Cham (2022)
8. Maize, E.: Contributions to the theory of error reduction in quasi-Monte Carlo methods. Ph.D. thesis, The Claremont Graduate School (1981)
9. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods. CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, Philadelphia (1992)
10. Sloan, I.H., Joe, S.: Lattice Methods for Multiple Integration. Oxford University Press, Oxford (1994)

Appendix A

The integral $I_{\cos}(d) := \int_0^\infty \cos(r)\exp(-r^2)\, r^{d-1}\mathrm{d}r$ and its relative, the integral $I_{\sin}(d) := \int_0^\infty \sin(r)\exp(-r^2)\, r^{d-1}\mathrm{d}r$, may be evaluated iteratively for $d = 1, 2, \ldots$. Note that using integration by parts:

$$\begin{aligned}
I_{\cos}(d) &= \cos(r)\exp(-r^2)\,\frac{r^d}{d}\bigg|_0^\infty \\
&\qquad - \int_0^\infty [\cos(r)(-2r)\exp(-r^2) - \sin(r)\exp(-r^2)]\,\frac{r^d}{d}\mathrm{d}r \\
&= \frac{2I_{\cos}(d+2) + I_{\sin}(d+1)}{d} \\
I_{\sin}(d) &= \sin(r)\exp(-r^2)\,\frac{r^d}{d}\bigg|_0^\infty \\
&\qquad - \int_0^\infty [\sin(r)(-2r)\exp(-r^2) + \cos(r)\exp(-r^2)]\,\frac{r^d}{d}\mathrm{d}r \\
&= \frac{2I_{\sin}(d+2) - I_{\cos}(d+1)}{d}
\end{aligned}$$

This implies that

$$\begin{aligned}
I_{\cos}(d+2) &= \frac{dI_{\cos}(d) - I_{\sin}(d+1)}{2}, \\
I_{\sin}(d+2) &= \frac{dI_{\sin}(d) + I_{\cos}(d+1)}{2},
\end{aligned}$$

Need to finish and check against VS Code today