

Chapter 2

Quasi-Monte Carlo for Fast Statistical Simulation of Circuits

2.1 Motivation

Continued device scaling has dramatically increased the statistical variability with which circuit designers must contend to ensure the reliability of a circuit to these variations. As discussed in the introduction to this thesis, traditional process corner analysis is no longer reliable because the variations are numerous and much more complex than can be handled by such simple techniques. Going forward, it is increasingly important that we account accurately for the statistics of these variations during circuit design. In a few special cases, we have analytical methods that can cast this inherently statistical problem into a deterministic formulation, e.g., optimal transistor sizing and threshold assignment in combinational logic under statistical yield and timing constraints, as in [MDO05]. Unfortunately, such analytical solutions remain rare. In the general case, some combination of complex statistics, high dimensionality, profound nonlinearity or non-normality, stringent accuracy, and expensive performance evaluation (e.g., SPICE simulation) thwart our analytical aspirations. This is where Monte Carlo methods [Gla04] come to our rescue as true statistical methods.

Monte Carlo simulation can emulate a real statistical process using a given technique for simulating any event from this statistical process. For example, the performance of chips coming out of a manufacturing process is emulated by simulating multiple instances of the relevant circuit, with each instance having a different set of values for its manufacturing related parameters. Over the years, Monte Carlo has become a standard technique for statistical simulation of circuits and for yield estimation during the design phase [SP81][HLT83][SKC99][Eli94]. How-

ever, we gain the flexibility and accuracy that Monte Carlo offers at the cost of speed: a single Monte Carlo run can cost a few thousand SPICE simulations. Given its importance, it is surprising that Monte Carlo has not received the research effort it deserves, from the EDA community. There has been much research for methods to either replace Monte Carlo simulation all together, using acceptance region modeling [AMH91][AGW94][SVK94], or replacing SPICE simulations, using response surface modeling [YKHT87][FD93][LLPS05]. The SiLVR model presented in Chap. 1 falls under the category of response surface models. However, these methods gain speed by sacrificing accuracy, and in many cases accuracy is very important. An ideal solution would be to somehow speed up Monte Carlo simulation, while still using SPICE simulations and maintaining the generality of its application. There has been some research on this [HLT83][SKC99], but there is much more that needs to be done.

Here we observe that similar problems exist in various other fields of science and engineering. In particular, we look at the field of computational finance, where pricing financial instruments and derivatives (e.g., Asian options, mortgage-backed securities) requires the simulation of high dimensional stochastic processes, for which Monte Carlo has remained the main practical method [Gla04]. These problems are not only very nonlinear, they can also be quite large: pricing a portfolio of options or securities over a several year horizon can create problems with 1,000+ statistical variables, as in [NT96b]. Accuracy is often required to the level of one basis point (a relative accuracy of 10^{-4}) under impressively short time constraints (minutes, in the case of real-time arbitrage). In this chapter we attempt at redeploying a particularly successful Monte Carlo technique from this domain to our problem domain of circuits. This technique is commonly referred to as *quasi-Monte Carlo* (QMC) and is essentially Monte Carlo, but using a *deterministic* set of points from some, so called, *low-discrepancy sequence*.

Note that the theoretical underpinnings of QMC are not completely new, as evidenced by number theoretic results by Halton in 1960 [Hal60a]. However, recent developments in both theory and implementation complexity, along with the empirical discovery that it is unexpectedly efficient at evaluating certain high-dimensional integrals, have propelled QMC onto the center stage in the computational finance world, as evidenced by extensive articles in both popular and practitioner literature (The Economist, August 12, 1995; The New York Times, September 25, 1995; Risk Magazine). QMC has also found application for high dimensional integration problems in physics [MC95][Spa95]. A motivational example from finance is provided by Ninomiya and Tezuka in [NT96b],

where they evaluate the price of a five-year discount bond. For this 1,439-dimensional problem, they observe a speedup of about 150 for an accuracy level of 1 basis point, on using QMC instead of Monte Carlo. Much work has been done to study the application of QMC to finance problems [CMO97][ABG98][OE04][PT95]. Our goal here will be to study its application to circuit problems.

In the rest of the chapter, we will review the standard Monte Carlo method and its convergence behavior, relevant results from number theory and resulting QMC techniques, and our proposed framework for applying QMC to statistical simulation of circuits. While developing our framework we will also discuss some important idiosyncrasies of the QMC technique, because of which a naive, direct application might not work well. Based on this discussion we will then develop the essential pieces of our flow. We shall see in the results section that this proposed framework can lead to speedups of about 2 to 50 times over standard Monte Carlo while maintaining the same level of accuracy. A concise version of this chapter was presented in [SR07b].

2.2 Standard Monte Carlo

Let us first concretely define the canonical problem that Monte Carlo simulation addresses. We take two seemingly very different examples to arrive at common terminology. We will then base further discussion on this common terminology and the canonical problem.

2.2.1 The Problem: Bridging Computational Finance and Circuit Design

Consider two problems from two completely different domains:

- A. pricing an Asian option in computational finance, and
- B. estimating circuit yield in VLSI design.

Let us see what is common between these two problems. This will allow us to develop a canonical representation for the general problem that Monte Carlo solves, which will further enable us to clearly understand and apply related results.

2.2.1.1 Pricing an Asian Option

An *option* gives an investor the right to purchase one unit of a security at a specified *strike price* K at a future time T ; for example, the right to purchase shares of company XYZ at 5 dollars per share on a fixed date in the future. Given K, T , we wish to determine the price that the investor should pay for this option at present time 0. Merton expanded

the work by Black and Scholes to develop the *Black-Scholes* options pricing model [Mer73]. Merton and Scholes received the Nobel Prize in Economics in 1973 for this and other related work. Among other results, the model gives the *payoff* on an arithmetic *Asian* option, one of several types of options [Gla04], as

$$\left[\frac{1}{T} \int_0^T S(t) dt - K \right]_+, \quad \text{where } [\cdot]_+ = \max(0, \cdot), \quad (2.1)$$

where $S(t)$ is the price of the underlying security (stock) at time t . $S(t)$ is given as

$$\frac{dS(t)}{dt} = rdt + \sigma x(t) \sqrt{dt} \quad (2.2)$$

$$\Rightarrow S(t) = S(0) e^{[(r-0.5\sigma^2)t + \sigma \int_0^t x(t) \sqrt{dt}]}, \quad (2.3)$$

where r is the *risk-free, continuously compounded interest rate*, and $x(t)$ is a random process such that for every instant t , $x(t) \sim \mathcal{N}(0, 1)$. Thus, $W(t) = \int_0^t x(t) \sqrt{dt}$ is a Wiener process [Gla04]; that is, $W(t) \sim \mathcal{N}(0, t)$. Here, $x(t) \sqrt{dt}$ embodies the random volatility in the price of the security and σ is the magnitude of this volatility.

The Black-Scholes model gives the appropriate price of the option at time 0 as the expected value of the *discounted* payoff:

$$K_0 = E \left\{ e^{-rT} \left[\frac{1}{T} \int_0^T S(t) dt - K \right]_+ \right\}, \quad (2.4)$$

where e^{-rT} accounts for the fact the option will be purchased at time 0, but exercised at a future time T . The typical way to evaluate this price K_0 is to first discretize time t into s samples, with equal steps of size Δt as

$$t_0 = 0, \quad \Delta t = \frac{T}{s}, \quad t_i = t_{i-1} + \Delta t, \quad i \in \{1, \dots, s\}. \quad (2.5)$$

Then, the $x_i = x(t_i)$ are s independent, identically distributed random variables $\sim \mathcal{N}(0, 1)$. We can now write the security price from (2.3) as

$$S(t_i) \approx S(0) e^{[(r-0.5\sigma^2)i\Delta t + \sigma \Delta t \sum_{j=0}^i x_j]}, \quad i = \{1, \dots, s\}. \quad (2.6)$$

Then, evaluating $S(t_i)$ at each time sample, we can numerically approximate the $\frac{1}{T} \int_0^T S(t) dt$ in (2.4) as

$$\frac{1}{T} \int_0^T S(t) dt \approx \bar{S}_s = \frac{1}{s} \sum_{i=1}^s S(t_i) \quad (2.7)$$

and compute the option price from (2.4) as

$$K_0 \approx E\{e^{-rT}[\bar{S}_s - K]_+\}. \quad (2.8)$$

Note that \bar{S}_s is a function of the s random variables $\{x_i\}_{i=1}^s$ that follow a joint multivariate density distribution $\pi(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \mathbf{I}_s)$. We can then write the option price as

$$K_0 \approx \int_{\mathbb{R}^s} f(\mathbf{x})\pi(\mathbf{x})d\mathbf{x}, \quad \text{where } f(\mathbf{x}) = e^{-rT}[\bar{S}_s - K]_+. \quad (2.9)$$

Hence, the problem is now of evaluating an integral over an s -dimensional space.

2.2.1.2 Estimating Circuit Yield

Consider some circuit with s statistical parameters, or simply *inputs*, $\{x_i\}_{i=1}^s$ and s_y performance metrics, or simply *outputs*, $\{y_i\}_{i=1}^{s_y}$. The relationship between the outputs and the inputs can be written as

$$\mathbf{y} = \mathbf{f}_{sim}(\mathbf{x}) \quad (2.10)$$

where evaluating \mathbf{f}_{sim} might involve running one or more circuit simulations (e.g., AC analysis) and subsequent computations to compute the metrics (e.g., gain), as needed to compute the metrics in \mathbf{y} . Of course, \mathbf{f}_{sim} also take the design variables as arguments, but we assume a fixed design for this discussion. Also, there are some specifications that the performance metrics must meet for an acceptable design. Denoting these specifications by $\{t_i\}_{i=1}^{s_y}$, we require $\{y_i \leq t_i\}_{i=1}^{s_y}$, or equivalently $\mathbf{y} \leq \mathbf{t}$. Please note that here we use \leq without any loss of generality. If for some given \mathbf{x} , the design meets this criterion, we denote the event as a *pass* event, otherwise it is a *fail* event. In the context of manufacturing variations, we might be interested in estimating the yield of the circuit given probability distributions for the statistical parameters. The yield is the percentage of manufactured instances of the circuit that pass the specifications. We now state this mathematically. Let us define \mathcal{A} , the *acceptance region* for a given design, as the set of input vectors that give us a passing circuit:

$$\mathcal{A} = \{\mathbf{x} : \mathbf{f}_{sim}(\mathbf{x}) \leq \mathbf{t}, \mathbf{x} \in \mathbb{R}^s\}. \quad (2.11)$$

Also, define the *characteristic function* of \mathcal{A} as

$$I_{\mathcal{A}}(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in \mathcal{A} \\ 0, & \mathbf{x} \notin \mathcal{A} \end{cases} \quad (2.12)$$

which is 1 for pass and 0 for fail. This is also known as the *indicator function* in the VLSI CAD literature [HLT83]. Now, we can define the circuit yield as the probability of a circuit instance lying in the acceptance region:

$$Y_t = P(\mathbf{x} \in \mathcal{A}) = E(I_{\mathcal{A}}(\mathbf{x})) \quad (2.13)$$

which can be written as

$$Y_t = \int_{\mathbb{R}^s} I_{\mathcal{A}}(\mathbf{x}) \pi(\mathbf{x}) d\mathbf{x}. \quad (2.14)$$

This is now a problem of s -dimensional integration, similar to the problem of pricing Asian options.

2.2.1.3 The Canonical Problem

Equations (2.9) and (2.14) are identical in their form

$$Q = \int_{\mathbb{R}^s} g(\mathbf{x}) \pi(\mathbf{x}) d\mathbf{x} \quad (2.15)$$

and suggest a canonical form for the general problem. Only one step remains before we can reach this canonical form. let π_i be the marginal probability density distribution for x_i and Π_i be the corresponding marginal cumulative distribution. Then, for independent x_i , we can write (2.15) as

$$\begin{aligned} Q &= \int_{\mathbb{R}^s} g(x_1, \dots, x_s) \pi_1(x_1) \pi_2(x_2) \dots \pi_s(x_s) d\mathbf{x} \\ &= \int_{[0,1]^s} g(\Pi_1^{-1}(z_1), \dots, \Pi_s^{-1}(z_s)) d\mathbf{z}, \end{aligned} \quad (2.16)$$

leading us to the canonical form we seek by renaming z_i as x_i :

$$Q = \int_{C^s} f(\mathbf{x}) d\mathbf{x}, \quad C^s = [0, 1]^s \quad (2.17)$$

where C^s is the unit cube in s dimensions. For the rest of our discussions in this chapter we consider only C^s as our integration domain and assume that all required transformations have been incorporated into the function f .

2.2.2 Monte Carlo for Numerical Integration: Some Convergence Results

The general integration problem does not usually admit an analytical solution. A common approach to solve it then is to use numerical integration or *quadrature*, also known as *cubature* for $s \geq 2$ [Str71][Coo99].

These quadrature rules typically involve evaluating the function f at strategically placed points in C^s and doing a weighted sum to arrive at the estimate for the integral Q . The problem with these classical cubature methods is that they become intractable as the dimensionality s increases. The following theorem [Nik50] states this problem concretely.

THEOREM 2.1 ([Nik50]). *Let $f \in W_p^k(C^s)$, where $W_p^k(C^s)$ is the Sobolev class [Ada75] of functions defined on the unit cube C^s whose weak derivatives up to order k exist and are bounded under the L_p norm (see Sect. 1.4 and [Ada75] for definitions). Let $Q(f)$ be the exact integral for f , and $Q_n^{det}(f)$ be any n -point quadrature approximation to $Q(f)$. If $pk > s$ then*

$$\inf_{Q_n^{det}} \sup_{\{f: \|f\|_{k,p} \leq 1\}} |Q(f) - Q_n^{det}(f)| = \Theta(n^{-k/s}) \quad (2.18)$$

where the norm $\|f\|_{k,p}$ is the norm for the Sobolev space $W_p^k(C^s)$.

The theorem essentially says that for a given class of smooth functions, the error of any numerical quadrature method using n deterministic points decreases asymptotically as $\Theta(n^{-1/s})$ with the dimensionality. This implies that to halve the error, the number of points must increase by a factor of 2^s . Also, to maintain the same error, the number of quadrature points must increase exponentially with the dimensionality s . Thus, for circuit yield estimation, the number of circuit simulations in (2.14) must increase exponentially with the number of statistical parameters. This can very easily become intractable, even for very few parameters. Here, we have run into the well-known *curse of dimensionality*. We face this “curse” in all the three chapters of this thesis and a part of each proposed method is some technique to defeat it.

Monte Carlo is able to defeat this curse. This is the primary reason for its popular adoption for computing high-dimensional integrals in a wide variety of fields. Another class of quadrature techniques based on sparse grids proposed by Smolyak [Smo63] also improves the convergence to make moderate-dimensional integration feasible. However, for large dimensions (100s) only Monte Carlo techniques are known to be tractable. For more details on sparse grid-based quadrature, please refer to [GG98]. The quadrature points used by standard Monte Carlo are randomly chosen. We will also refer to these as sample/sampling points in the context of Monte Carlo. In general, any random method for computation is a Monte Carlo method [Hei96], but we focus primarily on independent Monte Carlo, where every point is generated independently of the other points. Examples of dependent Monte Carlo methods are Markov chain Monte Carlo methods like Gibbs sampling and simulated

Algorithm 2.1 The standard Monte Carlo algorithm

Require: function f , joint probability distribution $\Pi(\mathbf{x})$, and sample size n

- 1: **for** $i = 1$ to n **do**
 - 2: randomly generate $\mathbf{x}_i = (x_1, \dots, x_s)$ from Π
 - 3: evaluate $y_i = f(\mathbf{x}_i)$
 - 4: **end for**
 - 5: **return** Monte Carlo estimate $Q_n = \frac{1}{n} \sum_{i=1}^n y_i$
-

annealing: a good survey is provided in [Fis06]. The standard Monte Carlo algorithm is shown as Algorithm 2.1.

One Monte Carlo run involves evaluating the function f at n randomly chosen locations in the input space. Since \mathbf{x}_i , and hence y_i , are independent and identically distributed, the Monte Carlo estimate Q_n converges almost surely to Q as the sample size n is increased by the strong Law of Large Numbers [HC71]; i.e.,

$$P(\lim_{n \rightarrow \infty} Q_n = Q) = 1. \quad (2.19)$$

From Algorithm 2.1, we can easily see that if we ran multiple n -point Monte Carlo runs, we would obtain a different estimate Q_n each time. As a result, the integration error of Monte Carlo is probabilistic in nature and a deterministic bound, as in Theorem 2.1, does not make sense. An *average* error, however, does make sense. Bakholev [Bak59] showed the following result.

THEOREM 2.2 (Bakholev [Bak59]). *Assume the conditions of Theorem 2.1. The average Monte Carlo integration error is $\Theta(n^{-\frac{k}{s}-\frac{1}{2}})$.*

A proof can be found in [Hei94]. Thus, we can significantly improve over the exponential complexity of the worst error for deterministic methods. For small s , the convergence behavior is close that for the classical quadrature methods, $n^{-\frac{k}{s}}$. However, for moderate to large values of s (typically ≥ 6), the dimension dependent part becomes negligible and the convergence is close to $n^{-\frac{1}{2}}$. The extra gain of $n^{-\frac{k}{s}}$ is possible if we exploit the smoothness of the function using variance reduction techniques: these are enhancements to the standard algorithm that reduce the variance of the estimate Q_n [Fis06]. If we do not exploit the smoothness, or if f is not necessarily smooth, we can still derive a similar result using standard statistics.

THEOREM 2.3. Let $f \in L_1(C^s)$ be integrable over C^s . Define

$$\sigma(f) = \left[\int_{C^s} (f(\mathbf{x}) - \bar{f})^2 d\mathbf{x} \right]^{\frac{1}{2}}, \quad \bar{f} = \int_{C^s} f(\mathbf{x}) d\mathbf{x} = Q. \quad (2.20)$$

Then the average (r.m.s.) error of Monte Carlo is

$$\sqrt{E[(Q - Q_n)^2]} \rightarrow \frac{\sigma(f)}{\sqrt{n}} \quad \text{as } n \rightarrow \infty. \quad (2.21)$$

PROOF. This is obvious from the central limit theorem (Theorem 3.2 in Sect. 3.2.2) [HC71], which says that

$$\lim_{n \rightarrow \infty} \frac{Q_n - Q}{\sqrt{\sigma^2/n}} \xrightarrow{d} \mathcal{N}(0, 1). \quad (2.22)$$

Hence, the Monte Carlo error decreases asymptotically as $n^{-\frac{1}{2}}$ for general integrable f . Note that the proportionality constant for this behavior is the standard target for variance reduction techniques like importance sampling, control-variates, and Rao–Blackwellization among others: for a review, see [Fis06][Gla04]. The next few sections will develop a framework that can improve on this convergence behavior, using quasi-Monte Carlo. Hence, it is complementary to these standard variance reduction techniques: it targets the behavior $n^{-\frac{1}{2}}$ and not the proportionality constant $\sigma(f)$.

2.2.3 Discrepancy: Uniformity and Integration Error

Suppose we have two different methods of numerical integration, which use the same number of points n , but the points are placed differently. We do not know anything else about the way these points are used by the two methods. Is there something we can say about the relative errors of the two methods with only this information regarding them?

One general way to address this question is to look at the properties of the quadrature point set being used, in particular the *uniformity* of the points. The following is based on Niederreiter’s development of this topic in the comprehensive [Nie78]. Before discussing this more theoretically, let us see an example to illustrate the context. Figure 2.1 shows two sets of points that might be used for integration, say by a Monte Carlo algorithm. In Fig. 2.1(a) we have a 200-point “random” sample generated using a standard pseudorandom number generator (e.g., the linear congruential generator [Fis06]). In Fig. 2.1(b) we have a 200-point

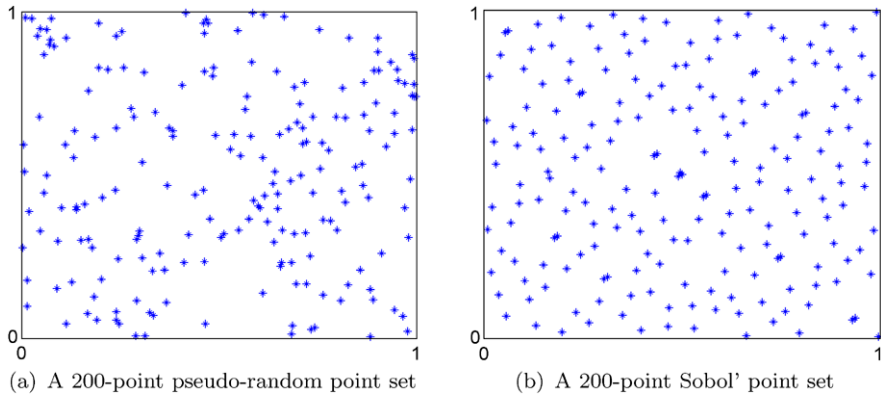


Figure 2.1. In two dimensions, Sobol' points are more uniformly distributed than typical pseudo-random points

“deterministic” sample from the so-called Sobol' sequence. It is immediately clear that the random sample is less *uniform* than the Sobol' sample. In other words there is more *discrepancy* in the way the random points are laid out from one region to the other, as compared to the Sobol' points. The uniformity, or rather the lack of it, is often measured in terms of a quantity understandably called *discrepancy*. Hence, we say that the points in Fig. 2.1(a) have high discrepancy, while those in Fig. 2.1(b) have low-discrepancy.

The uniformity of the point set is important because we are integrating over the entire domain C^s in (2.17), and the error will tend to 0 with increasing n only if the points are drawn from a uniform distribution over the entire unit cube. Hence, at least asymptotically the points should tend towards perfect uniform distribution over C^s . For a theoretical treatment of this intuitive explanation and a comprehensive discussion on uniformity please see [KN74]. The question is that if a point set achieves better uniformity (lower discrepancy) with some fixed finite n , is the corresponding integration estimate more accurate? We now review some theoretical results that try to address this question and suggest practical implications for Monte Carlo.

There can be several definitions for discrepancy [MC94][Hic98]. The one immediately relevant to our discussion is the L_∞ *star discrepancy*, or simply the *star discrepancy*, which we now define. The reader may use Fig. 2.2 as a reference illustration for the following. Let us say that we have n points $\{\mathbf{x}_i : \mathbf{x}_i \in C^s\}_{i=1}^n$ in our quadrature (Monte Carlo) point set. For some hyperrectangle $J \subseteq C^s$, let $\text{Vol}(J)$ be the volume of J and let $I_J(\mathbf{x})$ be the characteristic function (2.12) for J . Define the n_J as the

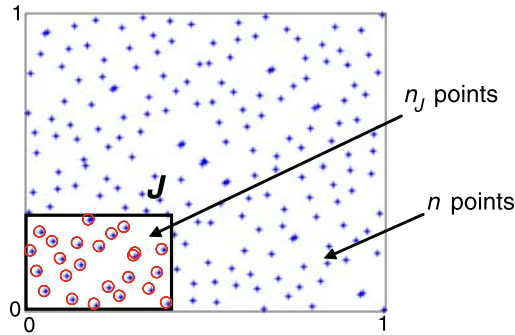


Figure 2.2. Illustration for the definition of discrepancy: n_J is the number of points inside any hyperrectangle J within the unit cube C^s

number of points lying inside J

$$n_J = \sum_{i=1}^n I_J(\mathbf{x}_i). \quad (2.23)$$

Then, we define the discrepancy as

$$D_n = \sup_{J \subseteq C^s} \left| \frac{n_J}{n} - \text{Vol}(J) \right|. \quad (2.24)$$

Hence, it is the maximum difference between the exact volume of any J and the estimate $(\frac{n_J}{n})$ of its volume using the points lying inside J . If we only look at hyperrectangles with one corner at the origin, $J = [\mathbf{0}, \mathbf{a})$ where $\mathbf{a} \in C^s$, then we get the star discrepancy

$$D_n^* = \sup_{\mathbf{a} \in C^s} \left| \frac{n_{[\mathbf{0}, \mathbf{a})}}{n} - \text{Vol}([\mathbf{0}, \mathbf{a})) \right|. \quad (2.25)$$

The following result by Koksma in one dimension and by Hlawka in multiple dimensions provides a partial, but useful answer to our question from the beginning of this section.

THEOREM 2.4 (Koksma–Hlawka [Hla61][Nie78]). *If function f has bounded variation in the sense of Hardy and Krause, then the Monte Carlo error is bounded as follows.*

$$\epsilon(f) = |Q - Q_n| \leq V(f) D_n^* \quad (2.26)$$

where $V(f)$ is the variation of f in the sense of Hardy and Krause, and D_n^* is the star discrepancy of the point set.

$V(f)$ is a measure of the total variation of the function over the unit cube. For a smooth function in one dimension

$$V(f) = \int_0^1 |df| \quad (2.27)$$

which is just the integral of the absolute value of the gradient of f . Hence, the more the function changes over the interval, higher is the value of $V(f)$. This can be generalized to multiple dimensions for non-smooth functions, in the sense of Hardy and Krause. The definition of this variation is not relevant for our discussion, and an intuitive understanding is sufficient. The definition is presented at the end of this section to avoid distraction.

Inequality (2.26) provides us an upper bound on the integration error for Monte Carlo using any given point set. It is particularly attractive because it separates out the two influences on the error: the properties of the function f , and the properties of the point set. Hence, it suggests that the error might be reduced if we used points with lower discrepancy. For a random sequence of points uniformly distributed over C^s , it has been shown that [Kie61]

$$D_n^* = O\left(\left[\frac{\log \log n}{n}\right]^{\frac{1}{2}}\right) \quad (2.28)$$

with probability 1. Combining this with the deterministic bound in (2.26) we see a good match with the $n^{-0.5}$ convergence of the probabilistic error bound (2.21) for standard Monte Carlo. Taking the suggestion of Theorem 2.4 we ask if there are point sequences with lower discrepancy, and does it help to replace random sampling with these sequences?

Similar to the star discrepancy, the L_2 star discrepancy is defined as

$$T_n^* = \left[\int_{C^s} \left(\frac{n_{[\mathbf{0}, \mathbf{a})}}{n} - \text{Vol}([\mathbf{0}, \mathbf{a})) \right)^2 d\mathbf{a} \right]^{\frac{1}{2}}. \quad (2.29)$$

It is known that [Nie78]

$$D_n^* \geq T_n^*. \quad (2.30)$$

Roth [Rot80] proved a lower bound for the L_2 star discrepancy of any set of n points in C^s , which then also applies to the star discrepancy

$$D_n^* \geq T_n^* > c_s \frac{(\log n)^{\frac{s-1}{2}}}{n} \quad (2.31)$$

where c_s depends only on s . For the first n points of an infinite sequence, it is modified [KN74] to

$$D_n^* > c'_s \frac{(\log n)^{\frac{s}{2}}}{n}. \quad (2.32)$$

We take this opportunity to clarify the difference between a set of n points and a sequence. The former is of finite size, while latter extends to infinite size. Hence, if the required sample size n is known before hand, we can better tailor the n point locations, as compared to when the required n is not known in advance and the point generation scheme must be able to keep generating points incrementally. The bound (2.31) applies to the former, while (2.32) applies to the latter. There is a widely believed conjecture in the theory of uniform distributions that says that a tighter bound exists with the exponent $\frac{s-1}{2}$ replaced by $s-1$ in (2.31) and $\frac{s}{2}$ replaced by s in (2.32). This has been proved only for $s \leq 2$ in the first case and for $s = 1$ in the second, but it is the best seen behavior yet for arbitrary s . Hence, any such sequence, for which

$$D_n^* = O\left(\frac{(\log n)^s}{n}\right) \quad (2.33)$$

is called a *low-discrepancy sequence* (LDS) or a *quasi-random sequence*. We will use the former term in this thesis. Halton, in [Hal60b], showed the existence of infinite deterministic sequences in any dimension s which satisfy (2.33), and provided a construction for one such sequence, commonly referred to as the Halton sequence. Hence, for large values of n , we can achieve n^{-1} convergence of the discrepancy, as compared to only $n^{-0.5}$ for random sequences. The points shown in Fig. 2.1(b) are from one such LDS, discovered by Sobol' [Sob67]. We can clearly see the lower discrepancy as compared to the pseudorandom points in Fig. 2.1(a).

Other definitions and generalization for discrepancy have been proposed [Nie78][MC94][Hic98][Wo91]. In many cases corresponding results similar to the Koksma–Hlawka inequality, often for some special class of functions, have been also provided. For example, [Wo91] provides an estimate for the *average* error over a class of functions following the Brownian sheet measure – a generalization of Brownian motion to s dimensions – using the L_2 star discrepancy T_n^* .

2.2.3.1 Variation in the Sense of Hardy and Krause

Here we define the variation of a function f over C^s in the sense of Hardy and Krause, as given in [Nie78]. For any interval (hyperrectangle) in C^s , $J = [a_1^{(1)}, a_2^{(1)}] \times \cdots \times [a_1^{(s)}, a_2^{(s)}] \subseteq C^s$, define

$$\delta(f; J) = \sum_{e_1=1}^2 \cdots \sum_{e_s=1}^2 (-1)^{e_1+\cdots+e_s} f(a_{e_1}^{(1)}, \dots, a_{e_s}^{(s)}). \quad (2.34)$$

Here we add up the function value at all “even” corners of J ($e_1 + \cdots + e_s$ even) and subtract out the function values at all the “odd” corners

($e_1 + \dots + e_s$ odd). Now define any grid over C^s with any number of slices along each dimension. Each slice along any dimension can be of any arbitrary, non-trivial width, but with no overlap between slices. The set of all single cells in the grid is a *partition* \mathcal{P} of C^s . Now define

$$V^{(s)}(f) = \sup_{\mathcal{P}} \sum_{J \in \mathcal{P}} |\delta(f; J)| \quad (2.35)$$

where the supremum is over all possible partitions of C^s . This is the variation in the sense of Vitali. Let $\mathbf{u} = \{i_1, \dots, i_k\}$ be a subset of the dimensions, such that $1 \leq k \leq s$ and $1 \leq i_1 < \dots < i_k \leq s$. Define $C_{\mathbf{u}}^s = \{\mathbf{a} \in C^s : a_i = 1 \text{ for } a_i \notin \mathbf{u}\}$ as the subset of C^s with all coordinates not in \mathbf{u} set to 1. With f restricted to $C_{\mathbf{u}}^s$, define $V^{(k)}(f; \mathbf{u})$ as the variation in the sense of Vitali over $C_{\mathbf{u}}^s$, where $k = \text{card}(\mathbf{u}) = |\mathbf{u}|$. Then, we can define $V(f)$ in the sense of Hardy and Krause as

$$V(f) = \sum_{k=1}^s \sum_{\{\mathbf{u}: |\mathbf{u}|=k\}} V^{(k)}(f; \mathbf{u}). \quad (2.36)$$

If $V(f)$ is finite, then f has bounded variation in the sense of Hardy and Krause. If f is sufficiently smooth (has finite partial derivatives of sufficient order), then we can use partial derivatives instead of the finite sums and differences in (2.35), as shown in [MC94].

2.3 Low-Discrepancy Sequences

Quasi-Monte Carlo is Monte Carlo performed with points from a deterministic low-discrepancy sequence (Sect. 2.2.3). There two main classes of LDS:

- 1) (t, s) -sequences, and
- 2) integration lattices.

(t, s) -sequences have enjoyed more popularity and research than integration lattices, one reason being that it is more difficult to extend lattices to infinite sequences. In this thesis, we focus on (t, s) -sequences for these reasons. The interested reader is referred to [HHLL00][FW94][HW81] for details on integration lattices.

2.3.1 (t, m, s) -Nets and (t, s) -Sequences in Base b

In this section we present a definition of (t, m, s) -nets and (t, s) -sequences in based b , following the development in [Nie87]. As a preview, we note that a (t, m, s) -net in base b is a *fixed* set of exactly b^m points, where b is the base we will work in (e.g., 2 if binary), and m determines the size

of this *finite* point set. Also, s is the number of dimensions and t is a measure of the quality of the sequence in terms of uniformity – smaller t will imply better uniformity for fixed m , s and b . These interpretations extend also to the case of (t, s) -sequences in base b , which are fixed *infinite* sequences of points in s dimensions, which are composed of an infinite number of (t, m, s) -nets in a particular manner. Smaller t still implies better uniformity, for fixed s and b . Now, we proceed towards a concrete definition.

A **b -ary box** is an interval of C^s of the form

$$J = \prod_{i=1}^s \left[\frac{a_i}{b^{d_i}}, \frac{a_i + 1}{b^{d_i}} \right) \quad (2.37)$$

for integers $d_i \geq 0$ and $0 \leq a_i < b^{d_i}$. Hence, if we create a grid over C^s with b^{d_i} slices of equal width along dimension i , then each cell of the grid is a b -ary box. If any $d_i = 0$, then there is no slice along dimension i . Given integers $b \geq 2$ and $0 \leq t \leq m$ we can define a **(t, m, s) -net** in base b as a point set consisting of b^m points, such that $n_J = b^t$ for every b -ary box with volume $\text{Vol}(J) = b^{t-m}$. We recall from (2.23) that n_J is the number of points lying inside J , as used in the definition of star discrepancy (2.25), which we reproduce here for convenience.

$$D_n^* = \sup_{\mathbf{a} \in C^s} \left| \frac{n_{[\mathbf{0}, \mathbf{a})}}{n} - \text{Vol}([\mathbf{0}, \mathbf{a})) \right|. \quad (2.38)$$

Figure 2.3 illustrates this idea with a $(0, 3, 2)$ -net in base 2: $t = 0$, $m = 3$, $s = 2$ and $b = 2$. The number of points in the net is $b^m = 2^3 = 8$. All possible 2-ary box shapes, with volume $b^{t-m} = 2^{0-3} = 1/8$ are shown cornered at the origin. Stacking any of these shapes side by side with no overlap, to fill out the unit square will give us all the 2-ary boxes with volume $1/8$ for that shape. Repeating this for all four shapes will give us all possible 2-ary boxes of volume $1/8$. We can see that every such 2-ary box contains exactly 1 ($b^t = 2^0$) point. Hence, we call this a $(0, 3, 2)$ -net in base 2, and we say that the net *balances* all 2-ary boxes with volume $1/8$. Any box J is balanced by a net with n points, if it contains exactly $n \times \text{Vol}(J)$ points; i.e., its volume can be exactly computed using the fraction of points lying in it ($\text{Vol}(J) = n_J/n$). This property of (t, m, s) -nets helps reduce the star discrepancy (2.38) by making the term in the supremum equal to zero for some choices of \mathbf{a} (for the b -ary boxes with $\text{Vol}(J) = b^{\tau-m}$, where $t \geq \tau \leq m$), and by reducing the chances of a large term for any \mathbf{a} . We note here that any (t, m, s) -net is also a (τ, m, s) -net for every integer $\tau \geq t$. By t we will imply the smallest such value of τ . We can see that smaller values of t lead to better uniformity, since the net can then balance, or uniformly fill, smaller boxes.

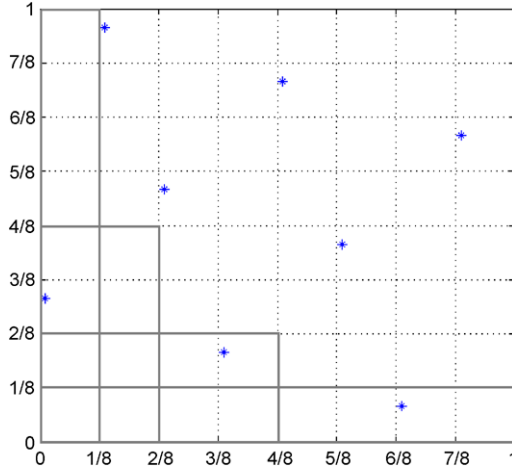


Figure 2.3. A $(0, 3, 2)$ -net in base 2 balancing all 2-ary boxes with volume $1/8$

For base b , we write $Z_b = \{0, 1, \dots, b-1\}$ for the set of digits in base b . For any real number $x \in [0, 1]$, we can write the b -adic expansion (e.g., binary expansion for base 2) as

$$x = \sum_{j=1}^{\infty} x_j b^{-j}, \quad x_j \in Z_b, \quad \forall j. \quad (2.39)$$

Truncating this at m digits (e.g., using 32 bits of computer precision), we define

$$[x]_{b,m} = \sum_{j=1}^m x_j b^{-j} \quad (2.40)$$

as the m -digit truncation of x . Here we make a sudden change in the notation for the coordinates of a vector: let us write any vector $\mathbf{x} \in C^s$ as $\mathbf{x} = (x^{(1)}, \dots, x^{(s)})$. This is done for notational convenience in the following theory. Then, we can write an m -digit truncation for \mathbf{x} in base b as

$$[\mathbf{x}]_{b,m} = ([x^{(1)}]_{b,m}, \dots, [x^{(s)}]_{b,m}). \quad (2.41)$$

Let $b \geq 2$ and $t \geq 0$ be integers. Then, we define a **(t, s) -sequence** in base b as a sequence of points $\{\mathbf{x}_i : i = \{1, 2, \dots\}\}$ in C^s , such that for all integers $k \geq 0$ and $m > t$, the set of points $\{[\mathbf{x}_i]_{b,m} : kb^m \leq i < (k+1)b^m\}$ is a (t, m, s) -net in base b . Again, we see that a sequence with smaller t for fixed s and b is preferred, since it will contain (t, m, s) -nets with smaller t . The popular constructions of Sobol' [Sob67] and Faure [Fau82] are instances of (t, s) -sequences as shown by Niederreiter [Nie87], who

then proposed (t, s) -sequences with better properties: Niederreiter sequences [Nie88] and, subsequently Niederreiter–Xing sequences [Nie98].

In the context of exploiting low discrepancy, as defined by (2.33), we want to know the discrepancy of (t, s) -sequences. According to [Nie87], the discrepancy of a (t, s) -sequence is bounded by

$$D_n^* \leq c(t, s, b) \frac{(\log n)^s}{n} + O\left(\frac{(\log n)^{s-1}}{n}\right), \quad \forall n \geq 2. \quad (2.42)$$

This shows that any (t, s) -sequence is an LDS as defined by (2.33). $c(t, s, b)$ is independent of n , and is given by

$$c(t, s, b) = \begin{cases} \frac{b^t}{s} \left(\frac{b-1}{2 \log b} \right)^s, & s = 2 \text{ or } b = 2, \quad s = 3, 4 \\ \frac{b^t}{s!} \frac{b-1}{2 \lfloor b/2 \rfloor} \left(\frac{\lfloor b/2 \rfloor}{\log b} \right)^s, & \text{otherwise} \end{cases}, \quad (2.43)$$

where $\lfloor x \rfloor$ is the greater integer $\leq x$. From this, we can see that smaller values of b are preferable for given t and s , as they lower the bound on the discrepancy. The Sobol' sequences [Sob67] are in the smallest base, $b = 2$, with t dependent on – and increasing with – s . It is also obvious from (2.43) that, for given b and s , smaller values of t are better, in agreement with our previous conclusion based on the definitions of (t, m, s) -nets and (t, s) -sequences. Faure's construction [Fau82] achieve the minimum value of t , $t = 0$, for b dependent on s : b is $p(s)$, the smallest prime $\geq s$.

Given this, it is natural to ask which of the two sequences is better. This is a difficult question and a clear answer is not known. In certain asymptotic terms, the discrepancy bound for the Faure sequences shows large improvement over the bound for the Sobol' sequences. The discrepancy bound constant $c(t, s, b)$ for Sobol' points takes the form

$$c_S = \frac{2^t}{s! (\log 2)^s}. \quad (2.44)$$

Sobol' [Sob67] gives the following bound for t as a function of s , for the Sobol' sequence:

$$t(s) \geq K \frac{s \log s}{\log \log s}. \quad (2.45)$$

This means that for the Sobol' sequences t increases superlinearly with increasing dimensionality s , and, thus, the constant in the discrepancy bound increases superexponentially with s . This is definitely not desirable. For the Faure sequence, the constant can be written as [Fau82]

$$c_F = \frac{1}{s!} \left(\frac{p(s) - 1}{2 \log p(s)} \right)^s, \quad (2.46)$$

which has the very desirable property that $\lim_{s \rightarrow \infty} C_F = 0$. However, we want to stress caution while using these *asymptotic* properties of the discrepancy *bounds* to compare the Sobol' and Faure sequences, or any other sequences. It is easily shown [MC94] that for practical values of n , the Faure bound actually increases to very large values before reducing back towards 0. We can see this by computing the maximum of the dominant term in the Faure bound. It is shown in [MC94], using basic calculus, that the maximum occurs for $n = e^s$. This shows that the dominant term in the Faure discrepancy bound increases with increasing number of points n up to $n = e^s$, whereas, it is to be expected that the actual discrepancy (uniformity) should reduce (improve) with increasing number of points. Hence, the convergence behavior of the bound sets in only after an extremely large number of points even for moderately large s .

Also, the bound in (2.42) is just that: a bound. There can be a large difference between the bound and the actual discrepancy in terms of magnitude and behavior. We cannot rely on the bound to compare the actual discrepancies of Sobol' and Faure points, as evidenced by the initial, but long increasing behavior of the Faure bound. These arguments extend for the general case of any set of (t, s) -sequences, and illustrate the difficulty in making a theory-backed choice between them. This difficulty is further worsened by the fact that the Koksma–Hlawka inequality (2.26) also only provides an upper *bound* on the integration error; and this bound may also be very loose, as shown for one class of functions in [Wo91][MC94]. [MC94] provides further insightful discussion and illustrations on this topic. Also see [Fox99], Chap. 12. We will choose the Sobol' points for our demonstrations, but based on empirical and practical considerations. However, this choice should not be taken as a definite rule for choosing Sobol' points over Faure points, since it is not backed by rigorous theoretical comparisons. We will revisit these considerations in more detail in Sect. 2.3.2.3, after we see how we can actually construct these (t, s) -sequences, along with some more examples that show better properties than both the Sobol' and Faure sequences.

2.3.2 Constructing Low-Discrepancy Sequences: The Digital Method

2.3.2.1 The Van der Corput Sequence: A Building Block

Van der Corput proposed one dimensional low-discrepancy sequences in 1935 [Van35], using b -adic expansions similar to (2.39) in some base b , where b is an integer ≥ 2 . Say we are generating the n -th point, where

n	n binary	$x_n = \psi_2(n)$ binary	x_n (fraction)
0	0	0. 0	0
1	1	0. 1	1/2
2	10	0. 01	1/4
3	11	0. 11	3/4
4	100	0. 001	1/8
5	101	0. 101	5/8
6	110	0. 011	3/8
7	111	0. 111	7/8
8	1000	0. 0001	1/16

Table 2.1. First nine points of a Van der Corput sequence in base 2

$n = 1, 2, \dots$. Consider the b -adic expansion of $n - 1$,

$$n - 1 = \sum_{k=0}^{\infty} a_k(n)b^k = \dots a_2a_1a_0, \quad (2.47)$$

where $a_k(n)$ is the k -th digit in the base b representation of $n - 1$, as represented by the last term, where we have dropped (n) to reduce notation. For finite n , only a finite number of $a_k(n)$ will be nonzero. The n -th Van der Corput point $x_n \in [0, 1)$ is then given by

$$x_n = \psi_b(n) = \sum_{k=1}^{\infty} \frac{a_{k-1}(n)}{b^k} = 0.a_0a_1a_2\dots \quad (2.48)$$

ψ_b is the *radical inverse function* and it basically mirrors the digits about the base b radix point. A example for base 2 is shown in Table 2.1. We can see how each subsequent point is strategically placed to fill out some largest remaining gap, ensuring good uniformity over the interval $[0, 1)$.

Halton [Hal60b] provided the first method for constructing an LDS in arbitrary dimensions, by extending Hammersley's method [Ham60] of generating finite point sets with low-discrepancy. The method uses one-dimensional Van der Corput sequences [Van35] with a distinct base for each coordinate, such that the bases are relatively prime integers greater than 1. Taking the first s prime numbers is typical since smaller bases result in uniformity with fewer samples for the Van der Corput sequence. However, the Halton sequence suffers from very poor uniformity in high dimensions because of an undesirable feature of the Van der Corput sequence for large base b . We illustrate this with the example of base 10. For $n = \{1, 2, 3, 4, 5\}$, the radical inverse function ψ_{10}

gives the first 5 points as $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ which are all clustered in one half of the interval $[0, 1)$. This “monotonic” filling is more pronounced for larger values of b , leading to only small parts of C^s being filled for even moderate dimensionality s . Hence, the Halton sequence is unsuited for our application where we expect to see s of the order of 10^1 – 10^2 , and we do not dwell further on it here. For more discussion on it and its improvements, which remain inadequate, see [Gla04]. The Sobol’ and Faure sequences were huge improvements over Halton’s construction and enabled practical use of QMC for large dimensions. Both these methods construct each coordinate by using some generalization of the Van der Corput sequence in some base that results in a permutation of the Van der Corput sequence. In fact, as we will see, both the Sobol’ and Faure constructions fall under a single general class of (t, s) -sequence constructions by Niederreiter [Nie92], called the *digital method*. The digital method uses generalizations of the Van der Corput sequence for generating the different coordinates of a sequence in s dimensions.

2.3.2.2 The Digital Method, Digital Nets and Digital Sequences

To avoid excessive technical notation, we now define digital nets and sequences only over a residue class field Z_b with b prime. The elements of Z_b are $\{0, 1, \dots, b-1\}$. Hence, we can define any real number in base b using this field. For a more general definition over arbitrary finite commutative rings, see [Nie92]. The reduced definition is more than sufficient for our endeavors.

Digital sequence: Let $s \geq 1$ be the dimensionality, and $b \geq 2$ be a prime base. Let $\{\mathbf{C}^{(i)}\}_{i=1}^s$ be $s \infty \times \infty$ matrices over Z_b ; i.e., each element of the matrices is a digit in base b . For integer $n \geq 1$ let

$$n - 1 = \sum_{k=0}^{\infty} a_k(n) b^k \quad (2.49)$$

be the base b representation of $n - 1$. Then define a sequence $\{\mathbf{x}_n\}$ with the n -th point

$$\mathbf{x}_n = (x_n^{(1)}, \dots, x_n^{(s)}), \quad (2.50)$$

$$x_n^{(i)} = \sum_{k=1}^{\infty} \frac{y_k^{(i)}(n)}{b^k}, \quad (2.51)$$

where $\{y_k^{(i)}(n)\}$ are given by

$$\begin{pmatrix} y_1^{(i)}(n) \\ y_2^{(i)}(n) \\ \vdots \end{pmatrix} = \mathbf{C}^{(i)} \cdot \begin{pmatrix} a_0(n) \\ a_1(n) \\ \vdots \end{pmatrix} \pmod{b}. \quad (2.52)$$

Such a sequence is called a *digital sequence* over Z_b . Here it is assumed that for each n only finitely many digits $(a_k(n))$ equal $b-1$. The matrices $\mathbf{C}^{(i)}$ are called *generator matrices*. Comparing with the Van der Corput sequence in (2.48) we see that each coordinate of this digital sequence is a permuted form of the Van der Corput sequence in base b . This permutation is provided by the generator matrices. For a $\mathbf{C}^{(i)} = \mathbf{I}_\infty$ we get the original base- b Van der Corput sequence.

Digital nets: We note that the generator matrices are of size ∞ to theoretically allow an infinite digit representation. However, in practice n will always be finite, in fact, only a few thousands or millions usually, needing only a finite number of significant digits in its b -adic expansion. If we use only m digits, then only the upper left $m \times m$ submatrix of every $\mathbf{C}^{(i)}$ will be relevant, and we can generate a maximum of b^m points. With such digit truncation, we are no longer truly generating a sequence with finite matrices; we are generating finite point sets or *nets*. A point set with b^m points, so generated, is called a *digital net*. In a practical setting, say while applying this digital method for yield estimation of circuits, we will set m sufficiently high such that we never generate b^m points. Also, we will typically need to have the ability to generate points incrementally, without initial knowledge of the exact value of n we will need. In such a case, even if we are use finite generator matrices, we are effectively choosing points from the underlying infinite *sequence*. Hence, it is sufficient and also more relevant, to discuss only sequences from here on.

Digital (t, s) -sequences: A digital sequence is of use to us here only when it is a (t, s) -sequence. Niederreiter [Nie92] provides us the criteria for this requirement. Let

$$\mathbf{c}_j^{(i)} = \{c_{j1}^{(i)}, c_{j2}^{(i)}, \dots\} \quad (2.53)$$

be the j -th row of matrix $\mathbf{C}^{(i)}$. For integer $m > 0$ let

$$\mathbf{c}_j^{(i)}(m) = \{c_{j1}^{(i)}, c_{j2}^{(i)}, \dots, c_{jm}^{(i)}\} \quad (2.54)$$

be the j -th row of the upper left $m \times m$ matrix of $\mathbf{C}^{(i)}$. Define a system of vectors

$$C = \{\mathbf{c}_j^{(i)} : 1 \leq j \leq m, 1 \leq i \leq s\}, \quad (2.55)$$

taking only the first m rows of the $\mathbf{C}^{(i)}$ matrices. Then for integers $0 \leq d_1, \dots, d_s \leq m$, and $\sum_{i=1}^s d_i = d$, define $\sigma(C)$ as the largest d such that any subsystem $\{\mathbf{c}_j^{(i)} : 1 \leq j \leq d_i, 1 \leq i \leq s\} \subseteq C$ is linearly independent over Z_b . Now define the system of vectors

$$C(m) = \{\mathbf{c}_j^{(i)}(m) : 1 \leq j \leq m, 1 \leq i \leq s\}. \quad (2.56)$$

For integers $m > t \geq 0$, if $\sigma(C(m)) \geq m - t$, then the corresponding digital sequence is a (t, s) -sequence in base b . Note that we are usually interested in the smallest such t .

Some notable examples of digital (t, s) -sequences are constructions by

- 1) *Sobol'* [Sob67]: Sobol' constructed (t, s) -sequences in base 2 for any dimension s , with t depending on s and of order of magnitude $O(s \log s)$. This leads to a superexponential increasing behavior for the constant $c(t, s, b)$ in the leading term of the discrepancy bound (2.42) for Sobol' sequences. This was discussed in more detail in Sect. 2.3.1. The generator matrices are constructed using the coefficients of primitive polynomials over the field Z_2 . A software implementation was shown in [BF88] and refined in [JK03]. We will describe the construction in detail in Sect. 2.3.3.
- 2) *Faure* [Fau82]: Faure constructed $(0, s)$ -sequences in any prime base $b \geq s$ for any dimension s . These sequences improved the asymptotic behavior of $c(t, s, b)$ in the discrepancy bound (2.42) to $\lim_{s \rightarrow \infty} c(t, s, b) = 0$. Implication of this "improvement" and related caveats were discussed in Sect. 2.3.1. Faure used powers of the upper-triangular Pascal matrix modulo b , \mathbf{P}_b to create the generator matrices:

$$\mathbf{C}^{(i)} = \mathbf{P}_b^{i-1}, \quad \text{for } i \geq 2, \quad \mathbf{C}^{(1)} = \mathbf{I} \quad (\text{identity matrix}). \quad (2.57)$$

The (j, k) -th element of the i -th generator matrix for $i \geq 2$ is then given by

$$c_{jk}^{(i)} = \begin{cases} b^{i-1} C_{j-1} i^{(k-j)}, & j \leq k \\ 0, & j > k \end{cases}, \quad i \geq 2. \quad (2.58)$$

See [Gla04] for further details. [Fox86] presents a software implementation.

- 3) *Niederreiter* [Nie88]: Niederreiter generalized these previous constructions and, for the first time, showed a construction for (t, s) -sequences for all dimensions s and all bases b . For fixed b , the order of magnitude of $t = O(s \log s)$, similar to Sobol' points. However, on using different b for different s , better values of t can be obtained.

The generator matrices are constructed using coefficients of distinct monic irreducible polynomials of minimum degree. A software implementation is presented in [BFN92]. A generalization of the Halton sequence using a polynomial version of the radical inverse function (Sect. 2.3.2.1) was shown by *Tezuka* [Tez93], which is similar to the Niederreiter sequences. [NT96b] shows how the Sobol' and Faure sequences are special cases of these sequences, and obtains natural generalizations for both Sobol' and Faure sequences.

- 4) *Niederreiter and Xing (NX)* [XN95]: These researchers proposed the idea of using algebraic curves over finite fields (or, equivalently global function fields) to construct generator matrices, resulting in (t, s) -sequences with significantly improved theoretical quality over all previous constructions. At least four different constructions on this idea were proposed by them and are summarized in [Nie98]. For any given s , the constructions in [XN95] and [NX96] achieve the lowest values of t . The best achievable order of magnitude of t for these NX sequences is $O(s)$ for fixed b , which is significantly better than the otherwise common $O(s \log s)$. [Pir02] shows an implementation of the construction in [XN95] for dimensions 4 to 16. The construction of NX sequences require algebraic curves with certain specific properties to achieve the optimal t [Nie98]. Known examples of such algebraic curves are limited, and this limits the number of dimensions that can be constructed. Further, due to the very abstract nature of the formulation of these constructions, it is difficult for the general practitioner to implement them.

2.3.2.3 Comparing (t, s) -Sequences and Choosing One

Table 2.2, reproduced from [Nie98], shows a comparison of the t values for (t, s) -sequences constructed in base b using methods 1 [Sob67], 3 [Nie88] and 4 [NX96]. The much lower t values for the NX sequences suggest much lower discrepancy (2.43) and, consequently and potentially, much lower integration error (2.26). However, as discussed above, there are significant implementation difficulties with the NX sequences. Also, the Niederreiter sequences (method 3) do not offer significant improvement over the Sobol' or Faure sequences, for the general case. Given these reasons and the immense popularity of the Sobol' and Faure sequences among practitioners, we make a choice between the latter two options, for our experiments. Theoretical considerations do not provide a clear choice, as discussed in Sect. 2.3.1. Hence, we rely on empirical observations provided in [ABG98][Gla04] which show better performance on using Sobol' sequences. Furthermore, the fact that the Sobol' sequences

s	1) Sobol' [Sob67]	3) Nied [Nie88]	4) NX [NX96]
1	0	0	0
2	0	0	0
3	1	1	1
4	3	3	1
5	5	5	2
6	8	8	3
7	11	11	4
8	15	14	5
9	19	18	6
10	23	22	8
11	27	26	9
12	31	30	10
13	35	34	11
14	40	38	13
15	45	43	15
16	50	48	15
17	55	53	18
18	60	58	19
19	65	63	19
20	71	68	21

Table 2.2. Comparison of values of t for (t, s) -sequences in base 2, for $1 \leq s \leq 20$. The NX sequences have the lowest t values, and the best uniformity properties in terms of discrepancy (2.42). Reproduced from [Nie98]

are in base 2 allows us to exploit fast bit-level Boolean operations in the software implementation. For these reasons, we will use Sobol' sequences as our representative LDS to demonstrate the performance of QMC, and we will discuss only their construction in detail. Note that the performance of Sobol' sequences can only be improved upon by using the significantly better NX points.

2.3.3 The Sobol' Sequence

Sobol' [Sob67] gave the first construction of a (t, s) -sequence (he used the name LP_τ -sequence). Here we review the construction in the context of the digital method. We first show how the generator matrices are constructed, after which we discuss some practical issues for optimizing the uniformity of the sequences and for fast software implementation. Since each coordinate in the sequence is generated using a distinct generator matrix, let us focus on only on dimension first, and it can then be easily extended to arbitrary dimensions. Dropping the superscript for dimension, we want to compute the generator matrix \mathbf{C} for a one dimensional

Sobol' sequence. We recollect that the sequence is in base 2, hence every element of \mathbf{C} is a bit: a 0 or a 1. In practice we will work with a finite number of bits for the generated values; say this is m . Then, \mathbf{C} is an $m \times m$ matrix. Each *column* of this matrix can be considered as an m -bit binary expansion of some number $v_j \in [0, 1)$, with an implied radix point: the uppermost element in the column is the most significant bit.

$$v_j = \sum_{k=1}^m \frac{c_{ki}}{2^k} = 0.c_{1i}c_{2i} \dots c_{mi}. \quad (2.59)$$

These numbers v_j are called *direction numbers*. Using these direction numbers, we can write the digital method of (2.52) as

$$x_n = a_0(n)v_1 \oplus a_1(n)v_2 \oplus \dots \oplus a_{m-1}(n)v_m, \quad n = 1, 2, \dots, \quad (2.60)$$

where \oplus denotes *bitwise* binary addition (modulo 2),

$$0 \oplus 0 = 0, \quad 0 \oplus 1 = 1, \quad 1 \oplus 0 = 1, \quad 1 \oplus 1 = 0, \quad (2.61)$$

which is the same as a *bitwise* XOR operation, and the a_i bits are from the binary representation of $n - 1$.

Now the problem is to compute the m direction numbers. Sobol's method starts by selecting a *primitive polynomial* over $Z_2 = \{0, 1\}$,

$$x^q + d_1x^{q-1} + \dots + d_{q-1}x + 1, \quad d_i \in \{0, 1\}, \quad \forall i. \quad (2.62)$$

This is a polynomial of degree q and coefficients d_i in $\{0, 1\}$, satisfying two properties with respect to binary arithmetic (modulo 2):

- it is irreducible; i.e., it cannot be factored, and
- the smallest power p for which the polynomial divides $x^p + 1$ is $p = 2^q + 1$.

Tables listing primitive polynomials are widely available, for example in [PW72], and generation algorithms have also been suggested, as in [RB95]. We also need to choose *odd* integers m_1, \dots, m_q , such that $0 < m_j < 2^j$. The polynomial (2.62) defines a recurrence relation,

$$m_j = 2d_1m_{j-1} \oplus 2^2d_2m_{j-2} \oplus \dots \oplus 2^{q-1}d_{q-1}m_{j-q+1} \oplus 2^qm_{j-q}, \quad j > q, \quad (2.63)$$

where again \oplus denotes bitwise binary addition (modulo 2), or bitwise XOR. Now we can define the direction numbers as

$$v_j = \frac{m_j}{2^j}. \quad (2.64)$$

Note that dividing by 2^j is equivalent to shifting the radix point to the left j places in the binary representation of m_j . Then, we can use (2.62) to define a recurrence relation for v_j ,

$$v_j = d_1 v_{j-1} \oplus c_2 v_{j-1} \oplus \cdots \oplus d_{q-1} v_{j-q+1} \oplus v_{j-q} \oplus \frac{v_{j-q}}{2^q}, \quad j > q. \quad (2.65)$$

Note that we are choosing the first q direction numbers by choosing the first q m_j values. The remaining $m - q$ direction numbers (columns of the generator matrix) can be computed using this recurrence relation.

We illustrate this procedure with an example. Consider the primitive polynomial

$$x^3 + x + 1, \quad (2.66)$$

where $q = 3$. Then the recurrence (2.64) becomes

$$v_j = v_{j-2} \oplus v_{j-3} \oplus \frac{v_{j-3}}{2^3}. \quad (2.67)$$

Suppose we initialize with $m_1 = 1, m_2 = 1, m_3 = 3$. The corresponding direction numbers are calculated by dividing m_j by 2^j , or shifting the binary radix point to the left by j places in the binary representation of m_j . Hence, in binary form

$$v_1 = m_1/2 = 0.1, \quad v_2 = m_2/2^2 = 0.01, \quad v_3 = m_3/2^3 = 0.011. \quad (2.68)$$

Also suppose that we are using $m = 5$ bits. Using the recurrence (2.67), we can compute the remaining $m - q = 5 - 3 = 2$ direction numbers:

$$\begin{aligned} v_4 &= v_2 \oplus v_1 \oplus \frac{v_1}{2^3} \\ &= 0.0100 \oplus 0.1000 \oplus 0.0001 \\ &= 0.1101, \\ v_5 &= v_3 \oplus v_2 \oplus \frac{v_2}{2^3} \\ &= 0.01100 \oplus 0.01000 \oplus 0.00001 \\ &= 0.00101. \end{aligned} \quad (2.69)$$

Using the bits of these direction numbers, we can write our generator matrix as

$$\mathbf{C} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \quad (2.70)$$

Note that the generator matrix is an upper diagonal matrix. This is true in general for any Sobol' generator matrix. This is because the j -th column (direction number) is generated by taking a number m_j with maximum j bits ($m_j < 2^j$). Also, every diagonal element is 1 because every m_j is odd. We can use this generator matrix in (2.52) to generate the Sobol' points. Instead, equivalently, we use (2.60) exploiting efficient bitwise binary operations.

$$\begin{aligned}
x_1 &= 0(0.10000) \oplus 0(0.01000) \oplus 0(0.01100) \oplus 0(0.11010) \oplus 0(0.00101) \\
&= 0.0 = 0 \\
x_2 &= \mathbf{1}(0.10000) \oplus 0(0.01000) \oplus 0(0.01100) \oplus 0(0.11010) \oplus 0(0.00101) \\
&= 0.1 = 1/2 \\
x_3 &= 0(0.10000) \oplus \mathbf{1}(0.01000) \oplus 0(0.01100) \oplus 0(0.11010) \oplus 0(0.00101) \\
&= 0.01 = 1/4 \\
x_4 &= \mathbf{1}(0.10000) \oplus \mathbf{1}(0.01000) \oplus 0(0.01100) \oplus 0(0.11010) \oplus 0(0.00101) \\
&= 0.11 = 3/4 \\
x_5 &= 0(0.10000) \oplus 0(0.01000) \oplus \mathbf{1}(0.01100) \oplus 0(0.11010) \oplus 0(0.00101) \\
&= 0.011 = 3/8 \\
x_6 &= \mathbf{1}(0.10000) \oplus 0(0.01000) \oplus \mathbf{1}(0.01100) \oplus 0(0.11010) \oplus 0(0.00101) \\
&= 0.111 = 7/8 \\
x_7 &= 0(0.10000) \oplus \mathbf{1}(0.01000) \oplus \mathbf{1}(0.01100) \oplus 0(0.11010) \oplus 0(0.00101) \\
&= 0.001 = 1/8 \\
&\vdots = \vdots
\end{aligned} \tag{2.71}$$

We can see that the resulting points are permutations of the Van der Corput sequence in base 2 (Sect. 2.3.2.1). For the case of multiple dimensions ($s > 1$), each dimension gets its own distinct primitive polynomial and a corresponding set of initial m_j values. This leads to different permutations of the Van der Corput sequence in different dimensions, resulting in uniform distribution in the sampling region C^s .

The Sobol' construction takes two external inputs for each dimension: the primitive polynomial and the set of m_j values. Two natural questions that follow are:

- How do these inputs affect the properties of the resulting sequence?
- What are good choices for these inputs?

Sobol' provides us with some answers to these questions. First we look at the choice of polynomials.

2.3.3.1 Choosing Primitive Polynomials for Good Sobol' Sequences

Sobol' [Sob67] showed that under certain conditions the t parameter for a Sobol' sequence is

$$t = \sum_{i=1}^s (q_i - 1) = q_1 + q_2 + \cdots + q_s - d, \quad (2.72)$$

where q_i is the degree of the primitive polynomial used for dimension i . In the general case, a Sobol' sequence might achieve a lower t value: this is an upper bound on the lowest t value, that is exact under the conditions given in [Sob67]. Since a lower value of t leads to better uniformity and a lower bound on the discrepancy (Sect. 2.3.1), this result recommends using polynomials of lowest possible degree. Hence, we sort the polynomials with nondecreasing degree and use them in the same order for increasing dimensions.

2.3.3.2 Choosing Initial Direction Numbers for Good Sobol' Sequences

Sobol' [Sob76] defines two uniformity properties for any sequences:

- Property A: An s -dimensional sequence $\{\mathbf{x}_n\}$ satisfies property A if for every $j = 0, 1, \dots$ exactly one of the points $\{\mathbf{x}_k : j2^s \leq k < (j+1)2^s\}$ falls in each of the 2^s cubes of the form

$$\prod_{i=1}^s \left[\frac{a_i}{2}, \frac{a_i + 1}{2} \right), \quad a_i \in \{0, 1\}. \quad (2.73)$$

In other words, every set of points $\{\mathbf{x}_k : j2^s \leq k < (j+1)2^s\}$ is a $(0, s, s)$ -net in base 2. Note that some similar properties are satisfied by any (t, s) -sequence (Sect. 2.3.1), but this property strengthens the uniformity requirement.

- Property A': An s -dimensional sequence $\{\mathbf{x}_n\}$ satisfies property A' if for every $j = 0, 1, \dots$ exactly one of the points $\{\mathbf{x}_k : j2^{2s} \leq k < (j+1)2^{2s}\}$ falls in each of the 2^{2s} cubes of the form

$$\prod_{i=1}^s \left[\frac{a_i}{4}, \frac{a_i + 1}{4} \right), \quad a_i \in \{0, 1, 2, 3\}. \quad (2.74)$$

In other words, every set of points $\{\mathbf{x}_k : j2^{2s} \leq k < (j+1)2^{2s}\}$ is a $(0, 2s, s)$ -net in base b . Once again, although there are similarities with the (t, m, s) -net properties of a (t, s) -sequence, this property strengthens the uniformity requirement.

Sobol' [Sob76] also provides conditions on the direction numbers to ensure these additional uniformity properties for the resulting Sobol' sequences. Denote the j -th direction number for the i -th dimension by $v_j^{(i)}$. Thus, the generator matrix $\mathbf{X}^{(i)}$ is composed from $\{v_1^{(i)}, v_2^{(i)}, \dots\}$, where we have used the column vector interpretation of each $v_j^{(i)}$. Denote the first bit of $v_j^{(i)}$ by $v_{j,1}^{(i)}$: this is also the first element of the j -th column of $\mathbf{C}^{(i)}$ or, equivalently, the j -th element of the first row of $\mathbf{C}^{(i)}$. Then, property A holds for the generated sequence if and only if

$$\begin{vmatrix} v_{1,1}^{(1)} & v_{2,1}^{(1)} & \dots & v_{s,1}^{(1)} \\ v_{1,1}^{(2)} & v_{2,1}^{(2)} & \dots & v_{s,1}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1,1}^{(s)} & v_{2,1}^{(s)} & \dots & v_{s,1}^{(s)} \end{vmatrix} \not\equiv 0 \pmod{2}. \quad (2.75)$$

Note that this condition is on the first s direction numbers. In practice we will use m direction numbers and for large s , m may be less than s . However, theoretically, all s direction numbers do exist from their recurrence relation (2.64). Following the notation from above, let us denote the *second* bit of $v_j^{(i)}$ by $v_{j,2}^{(i)}$: this is also the second element of the j -th column of $\mathbf{C}^{(i)}$ or, equivalently, the j -th element of the second row of $\mathbf{C}^{(i)}$. Sobol' also shows that property A' holds if and only if

$$\begin{vmatrix} v_{1,1}^{(1)} & v_{2,1}^{(1)} & \dots & v_{2s,1}^{(1)} \\ v_{1,2}^{(1)} & v_{2,2}^{(1)} & \dots & v_{2s,2}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1,1}^{(s)} & v_{2,1}^{(s)} & \dots & v_{2s,1}^{(s)} \\ v_{1,2}^{(s)} & v_{2,2}^{(s)} & \dots & v_{2s,2}^{(s)} \end{vmatrix} \not\equiv 0 \pmod{2}. \quad (2.76)$$

We note that property A applies to subsequences of length 2^s , whereas property A' applies to subsequences of length 2^{2s} . Hence, even for moderately large s (order of 10), property A is of more interest to us in practical settings. Bratley and Fox [BF88] provide values of m_j that satisfy property A, for up to 40 dimensions. Joe and Kuo [JK03] propose a method to compute good m_j values and further extend the list to 1,111 dimensions.

2.3.3.3 Gray Code Construction

Antanov and Saleev [AS79] show that the implementation of Sobol's construction is simplified if the binary representation $\{a_0(n), \dots, a_{m-1}(n)\}$ of $n-1$ in (2.60) is replaced by the Gray code representation $\{g_0(n), \dots, g_{m-1}(n)\}$ of $n-1$. They show that this does not affect the asymptotic discrepancy behavior of the sequence. The binary Gray code can be obtained from the binary representation using

$$g_{m-1} \dots g_1 g_0 = a_{m-1} \dots a_1 a_0 \oplus 0 a_{m-1} \dots a_1, \quad (2.77)$$

where a_i is the i -th significant bit in the binary representation and g_i is the corresponding i -th bit in the Gray code representation. The reason for the simplification is that the Gray code of subsequent integers $n-1$ and n differ only in one bit. Let us rewrite the Sobol' point x_n (2.60) in one dimension as

$$x_n = g_0(n)v_1 \oplus g_1(n)v_2 \oplus \dots \oplus g_{m-1}(n)v_m \quad (2.78)$$

using the Gray code of $n-1$. Suppose the Gray codes of $n-1$ ($\{g_i(n)\}$) and n ($\{g_i(n+1)\}$) differ in the l -th bit. Then, we can write

$$\begin{aligned} x_{n+1} &= g_0(n+1)v_1 \oplus g_1(n+1)v_2 \oplus \dots \oplus g_{m-1}(n+1)v_m \\ &= g_0(n)v_1 \oplus g_1(n)v_2 \oplus \dots \oplus (g_l(n) \oplus 1)v_l \oplus g_{m-1}(n)v_m \\ &= x_n \oplus v_l. \end{aligned} \quad (2.79)$$

Hence, the points can be computed recursively, using only one bitwise XOR operation instead of m in (2.60). In the next section, we take a diversion to review Latin hypercube sampling, which is a popular Monte Carlo sampling technique that also tries to ensure good uniformity, and has been suggested for use on circuit problems [SKC99].

2.3.4 Latin Hypercube Sampling

Latin hypercube sampling (LHS), introduced in [MBC79], is a variance reduction technique applied to Monte Carlo. Recalling the asymptotic Monte Carlo variance (2.21)

$$\sigma_{\text{MC}}^2 = \frac{\sigma^2(f)}{n}, \quad (2.80)$$

LHS reduces this variance by reducing the contribution of $\sigma^2(f)$. LHS is effective for functions that can be largely separated into a sum of one dimensional functions, each one depending on only one of the input variables. We will discuss this in more concrete terms after introducing the concepts of ANOVA decomposition and effective dimension in

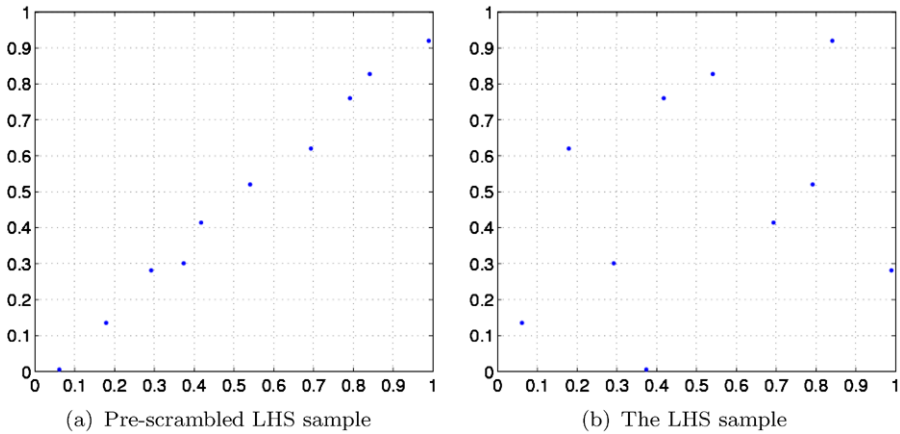


Figure 2.4. Latin hypercube sample of 10 points in 2 dimensions

Sect. 2.4.1. For now we review the construction of an LHS sample and satisfy ourselves with an intuitive, but convincing argument of its variance reduction effectiveness. We also discuss the connection between an LHS sample and (t, m, s) -nets.

2.3.4.1 Construction

Suppose we want to generate n points uniformly distributed in the s dimensional unit cube C^s . For this, a necessary condition is that the marginal distribution along each dimension should be uniform. Latin hypercube sampling tries to ensure good uniformity along each dimension as follows. Divide each dimension into n equal slices, or *strata*, forming a grid of n^s equal cells. For each stratum $j = \{1, \dots, n\}$ in dimension i , independently draw a uniformly distributed random value $y_j^{(i)}$ within the stratum. Generate n such values independently for each dimension:

$$y_j^{(i)} = \frac{j - 1 + U_j^{(i)}}{n}, \quad i = 1, \dots, s, \quad j = 1, \dots, n, \quad (2.81)$$

where $U_j^{(i)}$ are uniformly distributed, independent random variables over $[0, 1)$. This gives us n random values for each coordinate i , resulting in n points in C^s . However, the coordinate values for point j lie within the same stratum j along each dimension, resulting in points that are arranged in the diagonal cells. An example is shown in Fig. 2.4(a): the dotted lines show the strata along each dimension. This is definitely not uniformly distributed in the sense we desire. To achieve this uniform distribution, we randomly rearrange the strata along each dimension as

follows. For dimension i fix a permutation $\pi_i: \{1, \dots, n\} \rightarrow \{1, \dots, n\}$: this essentially “scrambles” the slices, and hence the coordinate values, along dimension i . s permutations π_1, \dots, π_s , one for each dimension, are randomly drawn from the set of $n!$ such permutations. Denote by $\pi_i(j)$ the permuted value of j : the j -th stratum along dimension i is scrambled to location $\pi_i(j)$. This scrambling of the strata along each dimension also causes a scrambling of the sampled coordinate values as

$$x_j^{(i)} = y_{\pi_i(j)}^{(i)} = \frac{\pi_i(j) - 1 + U_j^{(i)}}{n}, \quad i = 1, \dots, s, \quad j = 1, \dots, n, \quad (2.82)$$

with an independent scrambling scheme for each dimension. Now, we can compose the points in the LHS as

$$\mathbf{x}_j = \{x_j^{(1)}, x_j^{(2)}, \dots, x_j^{(s)}\}, \quad j = 1, \dots, n. \quad (2.83)$$

Figure 2.4(b) shows the resulting set of points after scrambling the points from Fig. 2.4(a). Note that the coordinate values of the points are not changed by this scrambling, only the relative ordering is. Hence, the points have very good uniformity along each dimension.

2.3.4.2 Variance (and Integration Error) Reduction

LHS is a special case of *stratified sampling* [Fis06][Gla04] – a popular, general method for variance reduction – because it stratifies each dimension. This stratification tries to ensure that the sample points are well spread out over the unit cube and there is not much variation in the way the integrand f is sampled if we generate different LHS samples with the same number of points. As a result there is also less variation in the integral estimate Q_n from one LHS run to another. Compare this with standard Monte Carlo, where due to lack of any such stratification, there is some chance that in two different runs the points will be clustered together in two different parts of the unit cube. This can result in large variation in the way f is sampled, and in the estimate Q_n . Hence, we often see a decrease in the variance of Q_n , and hence, the integration error, on using Latin hypercube sampling instead of standard Monte Carlo. McKay et al. [MBC79] derive the following result for the asymptotic variance of LHS.

$$\sigma_{\text{LHS}}^2 = \sigma_{\text{MC}}^2 + \frac{n}{n-1} \text{Cov}(\mu_1, \mu_2), \quad (2.84)$$

where μ_1, μ_2 are the mean values of f over any two cells in the grid resulting from the stratification, and Cov is the covariance, computed by taking the expectation over all possible pairs of cells. The paper

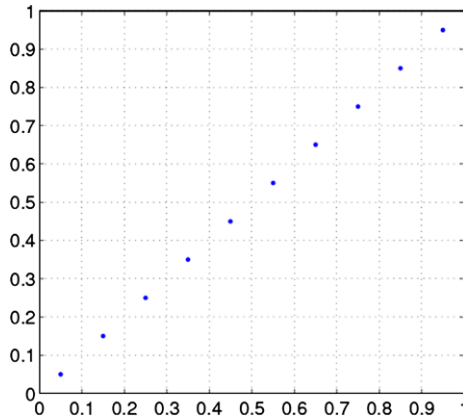


Figure 2.5. $(0,1,2)$ -net in base 10: pre-scrambled non-perturbation version of the LHS sample in Fig. 2.4(b)

also shows conditions when the second component of the variance can be negative, resulting in a variance reduction: when f is monotonic in each of its inputs. We review a different mathematical treatment of the variance reduction process of LHS, along with more general conditions on f that lead to efficient variance reduction, in Sect. 2.4.2.

2.3.4.3 LHS Sample Is a Scrambled (t, m, s) -Net

It is obvious from the construction that the sample size n is required in advance to generate an LHS sample, and arbitrary additions to the sample are not possible. One LHS run, then, generates a fixed set of points, also called a net as in our discussion of (t, m, s) -nets in Sect. 2.3.1. One popular construction of LHS samples replaces $U_j^{(i)}$ by $1/2$ in (2.82), placing every point at the exact center of the cell containing it. This improves the uniformity of the sample along each dimension, but increases the bias in the integral estimate: as the number of points is increased, Q_n does not tend exactly to Q . However, this error is often relatively small compared to the variance for practical sample sizes. The *pre-scrambled* version of such an LHS sample, with coordinate values given by

$$y_j^{(i)} = \frac{j - 1 + 0.5}{n}, \quad i = 1, \dots, s, \quad j = 1, \dots, n, \quad (2.85)$$

is a $(0, 1, s)$ -net in base n . This is because it is a set of n points that balances every n -ary box (refer (2.37)) of volume $n^{0-1} = 1/n$, as required by the definition of a $(0, 1, s)$ -net in base n . The resulting LHS sample is a *scrambled* $(0, 1, s)$ -net in base n , which has the same uniformity properties as the pre-scrambled version. The complete construction, us-

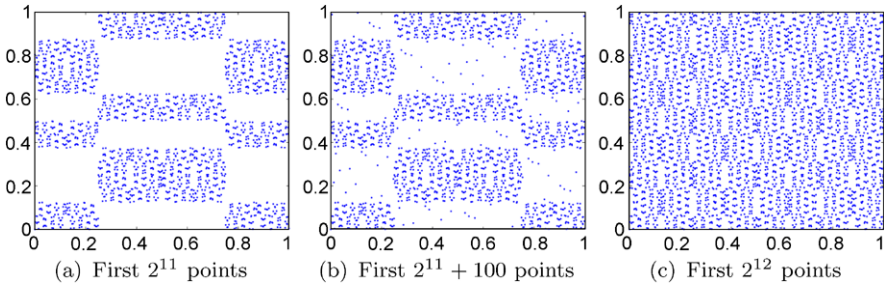


Figure 2.6. Dimensions 38 and 39 of a 40-dimensional Sobol' sequence showing undesirable patterns

ing $U_j^{(i)}$ as in (2.82) results in a scrambled $(0, 1, s)$ -net in base n with small random perturbations to the coordinates. The non-perturbed pre-scrambled $(0, 1, 2)$ -net in base 10 is shown in Fig. 2.5 for our two dimensional LHS example. We will further discuss scrambled nets and sequences in Sect. 2.5.3 in the context of extracting variance estimates for quasi-Monte Carlo. We will also revisit LHS in Sect. 2.4.2 where we compare it to Sobol' points.

2.4 Quasi-Monte Carlo in High Dimensions

A necessary, but not sufficient, condition for uniformity in all dimensions is uniformity in low dimension projections. Suppose we generate an $s = 40$ dimensional Sobol' point set. Now pick any two coordinates and plot the values of those coordinates. If we do not see a uniform distribution in the results space $[0, 1]^2$, then the point set is not uniform in 40 dimensions. Figure 2.6(a) plots coordinates 38 and 39, corresponding to primitive polynomials $x^8 + x^4 + x^3 + x^2 + 1$ and $x^8 + x^6 + x^5 + x^4 + 1$, respectively, for the first 2,048 Sobol' points. It is obvious that the projection is not uniform, and hence, the 40-dimensional Sobol' point set is not uniform. As we increase the number of points, the gaps get filled out and we achieve good uniformity in the projection, and ultimately in all dimensions.

We can see why this happens if we refer to the definition of a (t, s) -sequence in base b given in Sect. 2.3.1. Given $m > t$, we need at least b^{t+1} points for the (t, m, s) -net property to manifest. The minimum uniformity criterion for all dimensions in this context is to balance (equally fill) all b -ary boxes (2.37) resulting from the minimum number of non-trivial slices along each dimensions ($d_i > 0, \forall i$). The minimum number of slices is b , resulting in b^s boxes, each of volume b^{-s} . To balance these boxes with points from a (t, s) -sequence, a minimum of b^{t+s} points are

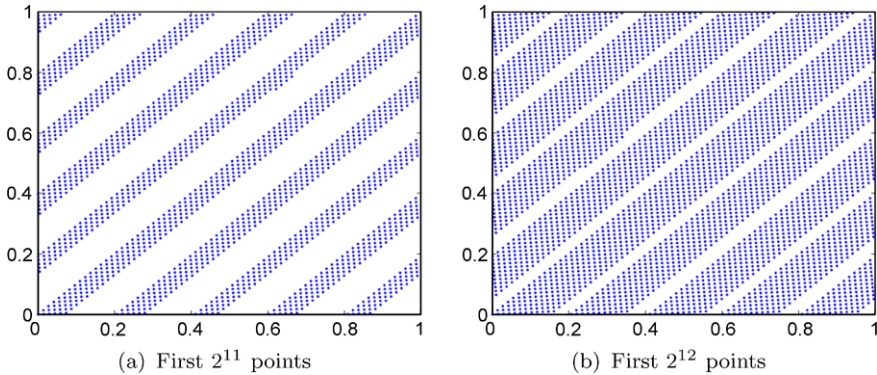


Figure 2.7. Dimensions 1 and 30 of a 70-dimensional Faure sequence showing undesirable patterns

required, such that every box has b^t points. The asymptotic discrepancy rate (2.42) therefore starts only from $n = b^{t+s}$ points. From Table 2.2 of t values, we can see that the required n can become astronomical for large s . For our Sobol' example, we use 8-th order polynomials for both dimensions 38 and 39. From the t value bound of (2.72), this 2 dimensional projection has a t value of $8 + 8 - 2 = 14$ or less. Hence, we should expect to see good uniformity with 2^{14} points. If we add the next 100 points, we see that the gaps start to get filled in Fig. 2.6(b), and we actually reach good uniformity with 2^{12} points (Fig. 2.6(c)), indicating that the t value for this two dimensional projection might be less than 14. We see similar patterns with other low discrepancy sequences too; Fig. 2.7 shows an example for a sample from a Faure sequence [Fau82].

Lack of uniformity in high dimensions is also evidenced by the asymptotic discrepancy behavior

$$D_n^* = O\left(\frac{(\log n)^s}{n}\right). \quad (2.86)$$

For large s , the numerator dominates for practical values of n . An extremely large number of samples is required for the denominator to dominate and for the rate to improve to n^{-1} . Despite this impractically large values of n , QMC has been found to work well for several high-dimensional problems in finance [NT96b][PT95][Gla04] with realistic sample sizes. The reason for this is that low dimensional projections of QMC points can have small discrepancy, and this can exploit dominant low dimensional structure of the integrand. For example, the two dimensional projection of Fig. 2.6 has a t value of ≤ 14 , while the original 40 dimensional sequence has a much larger t value (≤ 193). We next re-

view some theoretical concepts that provide us guidelines for exploiting this feature.

2.4.1 Effective Dimension of the Integrand

Assume function $f \in L_2(C^s)$ is square integrable over C^s . Let $\mathbf{u} \subseteq \{1, \dots, s\}$ denote any subset of the input dimensions of f . We use $|\mathbf{u}|$ for its cardinality and $-\mathbf{u}$ for its complementary set $\{1, \dots, s\} - \mathbf{u}$. Then, for any point $\mathbf{x} = \{x_1, \dots, x_s\} \in C^s$, $\mathbf{x}_{\mathbf{u}} = \{x_i : i \in \mathbf{u}\}$ is the vector of the coordinates of \mathbf{x} belonging to \mathbf{u} . $C^{\mathbf{u}}$ is the unit cube in the dimensions belonging to \mathbf{u} . We can write f as the sum of 2^s “simpler” functions using its *analysis of variance* (ANOVA) decomposition as

$$f(\mathbf{x}) = \sum_{\mathbf{u} \subseteq \{1, \dots, s\}} f_{\mathbf{u}}(\mathbf{x}), \quad (2.87)$$

where each function $f_{\mathbf{u}}$ depends only on $\mathbf{x}_{\mathbf{u}}$, excluding the effect of the proper subsets of $\mathbf{x}_{\mathbf{u}}$. We note that the integral of f over $-\mathbf{u}$ is a function of only $\mathbf{x}_{\mathbf{u}}$. For example, if $s = 2$, $f = x_1^2 + x_2^3$, $\mathbf{u} = \{1\}$ and $-\mathbf{u} = \{2\}$,

$$\int_{C^{\{2\}}} (x_1^2 + x_2^3) dx_2 = x_1^2 + 1/4. \quad (2.88)$$

Hence, the ANOVA terms are defined as

$$f_{\mathbf{u}}(\mathbf{x}) = \int_{C^{-\mathbf{u}}} \left(f(\mathbf{x}) - \sum_{\mathbf{v} \subset \mathbf{u}} f_{\mathbf{v}}(\mathbf{x}) \right) d\mathbf{x}_{-\mathbf{u}} \quad (2.89)$$

$$= \int_{C^{-\mathbf{u}}} f(\mathbf{x}) d\mathbf{x}_{-\mathbf{u}} - \sum_{\mathbf{v} \subset \mathbf{u}} f_{\mathbf{v}}(\mathbf{x}). \quad (2.90)$$

To compute the function $f_{\mathbf{u}}$ we subtract out the effect of all the proper subsets of \mathbf{u} and then average over the dimensions that are not in \mathbf{u} . Note that for the empty set, $f_{\emptyset}(\mathbf{x}) = \int_{C^s} f(\mathbf{x}) d\mathbf{x} = Q$ is a constant equal to the integral of f . These ANOVA terms enjoy the following properties.

- $\int_0^1 f_{\mathbf{u}} dx_j = 0$ for any $j \in \mathbf{u}$.
- The ANOVA decomposition is orthogonal: $\int_{C^s} f_{\mathbf{u}} f_{\mathbf{v}} d\mathbf{x} = 0$ if $\mathbf{u} \neq \mathbf{v}$.
- If $\sigma^2 = \int_{C^s} (f(\mathbf{x}) - Q)^2 d\mathbf{x}$ is the variance of f , then $\sigma^2 = \sum_{|\mathbf{u}| > 0} \sigma_{\mathbf{u}}^2$, where $\sigma_{\mathbf{u}}^2 = \int_{C^s} f_{\mathbf{u}}(\mathbf{x})^2 d\mathbf{x}$ is the variance of $f_{\mathbf{u}}$. Note that $\sigma_{\emptyset} = 0$ since $f_{\emptyset}(\mathbf{x})$ is constant.

We can use the variance contribution $\sigma_{\mathbf{u}}^2$ of any $f_{\mathbf{u}}$ to measure the relative importance of $f_{\mathbf{u}}$. In fact, normalized variances $\sigma_{\mathbf{u}}^2/\sigma^2$ are used as such

measures and are called *global sensitivity indices* in [SK05]. These are similar, but more general, in concept to the relative global sensitivity metric proposed in Sect. 1.6.3.1 of this thesis.

We now make some observations regarding f using its ANOVA decomposition. Let $g_t(\mathbf{x}) = \sum_{|\mathbf{u}|=t} f_{\mathbf{u}}(\mathbf{x})$ for $0 \leq t \leq s$. Then g_t captures that part of f that depends on t dimensional inputs, or, in other words, the part that is exactly t dimensional. Consequently, $\sum_{i=1}^t g_i(\mathbf{x})$ is that part of f that is at most t dimensional. From the orthogonality of the ANOVA terms, it follows that the variance of g_t is $\sigma^2(g_t) = \sum_{|\mathbf{u}|=t} \sigma_{\mathbf{u}}^2$. If, for some f , $\sigma^2(g_1) \geq 0.99\sigma^2$, then most (99%) of the variance of f is contributed by one dimensional ANOVA terms, and we say that f is effectively one dimensional in its inputs. Similarly, if $\sigma^2(g_1) + \sigma^2(g_2) + \sigma^2(g_3) \geq 0.99\sigma^2$, then f is effectively three dimensional in its inputs. Caffish et al. [CMO97] formalize two measures of the *effective dimension* of any function f .

Superposition sense: *The effective dimension of f with variance σ^2 , in the superposition sense, is the smallest integer s_S such that*

$$\sum_{0 < |\mathbf{u}| \leq s_S} \sigma_{\mathbf{u}}^2 \geq 0.99\sigma^2. \quad (2.91)$$

Truncation sense: *The effective dimension of f with variance σ^2 , in the truncation sense, is the smallest integer s_T such that*

$$\sum_{\mathbf{u} \subseteq \{1, \dots, s_T\}} \sigma_{\mathbf{u}}^2 \geq 0.99\sigma^2. \quad (2.92)$$

From these definitions, we can see that $s_S \leq s_T$. Wang and Fang [WF03] extend Sobol's method of computing ANOVA variances [SK05], to compute effective dimension. The proposed technique uses extensive Monte Carlo runs to approximate the variances of the ANOVA terms. The threshold of 0.99 in the definitions is arbitrary; other values may be preferable in different setting. s_S is an indicator of whether only low dimensional interactions dominate the variance in f , while s_T is the number of leading dimensions, given an ordering, that account for most of the variance in f . For example, if $f = x_1 + x_2 + x_4$, only one dimensional "interactions" can be added up to explain f . Hence, $s_S = 1$. However, the four leading dimensions $\{x_1, \dots, x_4\}$ are needed to explain at least 99% of the variance of f . Hence, $s_T = 4$. Note that if we reorder the dimensions by swapping x_3 and x_4 , then only the leading 3 dimensions are needed and s_T is now reduced to 3. We will rely on such reorderings to make QMC effective even in high dimensions for our circuit analysis problems. We note here that, typically, circuit performance metrics

are significantly affected only by some small subset of the parameters in the circuit. This claim is supported by results from experiments on the SiLVR model proposed in Chap. 1 of this thesis. These results are presented in Sect. 1.7. Hence, in many cases, the integrand for a statistical circuit analysis problem (e.g., Sect. 2.2.1.2) has a low effective dimension, at least in the truncation sense, assuming a proper ordering of the statistical parameters. The following result from [CMO97] provides hints as to how we can exploit this feature of our integrands.

For an n -point sample from an LDS, let $D_{n,\mathbf{u}}^*$ be the star discrepancy of the $|\mathbf{u}|$ dimensional points obtained by selecting only the coordinates in \mathbf{u} . For example the discrepancy of the two dimensional projection of the Sobol' points on $\mathbf{u} = \{38, 39\}$ is denoted by $D_{n,\{38,39\}}^*$. Denote the integration error of an n -point quadrature on any function f by

$$e_n(f) = \left| \int_{C^s} f(\mathbf{x}) d\mathbf{x} - \frac{1}{n} \sum_{i=1}^n f(\mathbf{x}_i) \right|, \quad (2.93)$$

where $\{\mathbf{x}_i\}_{i=1}^n$ are the quadrature points: the sample points for Monte Carlo or QMC. We can write this error using the ANOVA terms of f as

$$\begin{aligned} e_n(f) &= \left| \int_{C^s} \sum_{\mathbf{u} \subseteq \{1, \dots, s\}} f_{\mathbf{u}}(\mathbf{x}) d\mathbf{x} - \frac{1}{n} \sum_{i=1}^n \sum_{\mathbf{u} \subseteq \{1, \dots, s\}} f_{\mathbf{u}}(\mathbf{x}_i) d\mathbf{x} \right| \\ &= \left| \sum_{\mathbf{u} \subseteq \{1, \dots, s\}: |\mathbf{u}| > 0} \left[\int_{C^{\mathbf{u}}} f_{\mathbf{u}}(\mathbf{x}_{\mathbf{u}}) d\mathbf{x}_{\mathbf{u}} - \frac{1}{n} \sum_{i=1}^n f_{\mathbf{u}}((\mathbf{x}_i)_{\mathbf{u}}) \right] \right| \\ &\quad (\text{using } f_{\emptyset}(\mathbf{x}) = \text{constant } Q) \\ &\leq \sum_{\mathbf{u} \subseteq \{1, \dots, s\}: |\mathbf{u}| > 0} \left| \int_{C^{\mathbf{u}}} f_{\mathbf{u}}(\mathbf{x}_{\mathbf{u}}) d\mathbf{x}_{\mathbf{u}} - \frac{1}{n} \sum_{i=1}^n f_{\mathbf{u}}((\mathbf{x}_i)_{\mathbf{u}}) \right| \\ &\quad (\text{using the triangle inequality}) \\ &= \sum_{\mathbf{u} \subseteq \{1, \dots, s\}: |\mathbf{u}| > 0} e_n(f_{\mathbf{u}}). \end{aligned} \quad (2.94)$$

Then, using the Koksma–Hlawka inequality (2.26) for each ANOVA term, we can write

$$e_n(f) \leq \sum_{\mathbf{u} \subseteq \{1, \dots, s\}: |\mathbf{u}| > 0} e_n(f_{\mathbf{u}}) \leq \sum_{\mathbf{u} \subseteq \{1, \dots, s\}: |\mathbf{u}| > 0} V_{\mathbf{u}}(f_{\mathbf{u}}) D_{n,\mathbf{u}}^*, \quad (2.95)$$

where $V_{\mathbf{u}}(f_{\mathbf{u}})$ is the variation of $f_{\mathbf{u}}$ taken as a function over $C^{\mathbf{u}}$. Different versions of this relation are given in [WS07]. Also see [Hic98]. If,

for the subsets \mathbf{u} that show large variance $\sigma_{\mathbf{u}}$ (related to $V_{\mathbf{u}}(f_{\mathbf{u}})$), the discrepancy $D_{n,\mathbf{u}}^*$ of the projection of the LDS onto \mathbf{u} is small, then all the terms in the error bound are small, leading to a small error bound. This suggests two possible ways of achieving low integration error in high dimensions with QMC points that are not very uniform in high dimensions. In both these ways we exploit any low effective dimension properties of the integrand, for example, the integrand for circuit yield analysis.

- 1) If the effective dimension of f in the superposition sense, s_S , is small, it may be possible achieve very low integration error with an LDS that has good uniformity in low dimensional projections. From the discussion at the beginning of this Sect. 2.4, we know that QMC points can achieve good uniformity in low dimensional projections. This may not be true for some combinations of dimensions, as shown in Fig. 2.6, but on average the low dimension projections can show better uniformity than pseudorandom points even for realistic sample sizes [WF03][WS07]. In other words, for small $|\mathbf{u}|$, $D_{n,\mathbf{u}}^*$ is often small. If s_S is small then the high variance ANOVA terms are functions on subsets \mathbf{u} with small $|\mathbf{u}|$. Hence, from (2.95), this can lead to low integration errors, even if the overall discrepancy (2.86) of the LDS is large.
- 2) If the effective dimension of f in the truncation sense, s_T , is small, we can lower the error bound with an LDS that is uniform in the first s_T dimensions, even if the higher dimensions are not sampled uniformly. Typically, the initial dimensions are sampled more uniformly than the higher dimensions by samples from an LDS. Hence, $D_{n,\mathbf{u}}^*$ can be small for $\mathbf{u} = \{1, \dots, s_T\}$ if s_T is not too large. We can see this for the case of Sobol' points, where the early dimensions are generated using the lowest degree primitive polynomials. Hence, using (2.72) the t value of the sequence containing only the first s_T dimensions (e.g., dimensions 1–10) will be lower than the t value of any sequence containing any higher s_T dimensions (e.g., dimensions 91–100). For circuit yield problems, low s_T can be achieved for the integrand by arranging the statistical parameters in decreasing order of their impact on the relevant circuit performance, assuming that the total number of important parameters is not large.

Researchers in finance use linear transformations such as *Brownian bridge* (BB) and *principal components analysis* (PCA) on the input variables to reduce the truncation dimension: most of the variance in the resulting joint probability distribution of the transformed inputs is concentrated in the early dimensions. Extensive experiments showing

the advantage of exploiting reduced effective dimension can be found in [MC96][CMO97][ABG98][WF03][Owe03b]. For example, [WF03] shows that for the problem of pricing an Asian option, the truncation dimension can be reduced from 53 to 2 using PCA for a 64 dimensional problem, resulting in large reductions in error and much improved convergence. PCA is widely used in the electronic design automation community for reducing the number of statistical parameters to a few dominant ones that explain most of their variance [Ism93][CS05][LLPS05][LLP04]. In our proposed framework, we start from the result of any such PCA, and must further reduce the truncation dimension without the option of using PCA. Hence, we skip a detailed explanation. For details regarding PCA and BB, please refer to [Gla04].

2.4.2 Why Is Quasi-Monte Carlo (Sobol' Points) Better Than Latin Hypercube Sampling?

We saw in Sect. 2.3.4 that an LHS sample is essentially a scrambled (t, m, s) -net, specifically a $(0, 1, s)$ -net in base n , where n is the sample size. Hence, it is natural to ask if we gain any improvement by moving to a more general QMC approach, say using Sobol' sequences? If yes, then what are the reasons for such improvement? With the knowledge of ANOVA decomposition and effective dimension, we now address these questions, and provide simple illustrative examples in the results section, Sect. 2.6.

We know that LHS is able to maintain very good uniformity in all one dimensional projections because of its per-dimension stratification scheme. As a result of this, the variance error in integrating the one dimensional ANOVA terms $\{f_{\mathbf{u}} : |\mathbf{u}| = 1\}$ is very small. Using the orthogonality of ANOVA decomposition, we can write the overall function variance as

$$\sigma^2 = \sigma_1^2 + \sigma_{>1}^2: \quad \sigma_1^2 = \sum_{|\mathbf{u}|=1} \sigma_{\mathbf{u}}^2, \quad \sigma_{>1}^2 = \sum_{|\mathbf{u}|>1} \sigma_{\mathbf{u}}^2, \quad (2.96)$$

where σ_1^2 is the variance of the one dimensional part of f given by $g_1(\mathbf{x}) = \sum_{|\mathbf{u}|=1} f_{\mathbf{u}}(\mathbf{x})$ and $\sigma_{>1}^2$ is the remaining variance of $f - g_1$. Also write the asymptotic variance of the standard Monte Carlo estimate as

$$\sigma_{\text{MC}}^2 = \frac{\sigma^2}{n} = \frac{\sigma_1^2 + \sigma_{>1}^2}{n}. \quad (2.97)$$

Using ANOVA decomposition, Stein [Ste87] showed that the asymptotic variance of an LHS estimate is

$$\sigma_{\text{LHS}}^2 = \frac{\sigma_{>1}^2}{n} + o\left(\frac{1}{n}\right). \quad (2.98)$$

Hence, compared to the Monte Carlo estimate (2.97), LHS achieves a variance reduction by reducing the variance in estimating the integral of the one dimensional part of f to $o(n^{-1})$.

Note that every one dimensional projection of the LHS sample is a scrambled $(0, 1, 1)$ -net in base n . As a result, if the one dimensional part is smooth (the derivatives of $f_{\mathbf{u}}$ for $|\mathbf{u}| = 1$ are continuous) then the second term on the right hand side of (2.98) reduces as $O(n^{-3})$, as per the results of Owen [Owe97b] regarding scrambled nets (see (2.104)). It is clear from (2.98) that this reduction is effective only if f has significant variance contribution from its one dimensional component g_1 . If f has an effective dimension of 1 in the superposition sense ($s_T = 1$) then LHS is an excellent quadrature technique. Even if $s_T > 1$, many integrands have large variance contribution from their one dimensional components, explaining the success of LHS as a variance reduction technique. This result also explains why LHS is *unsuccessful* as a variance reduction technique in many settings: the integrand in those cases is probably not primarily one dimensional, because of which the gains over standard Monte Carlo are minimal.

Based on (2.98), for reasonably large sample size n , we can assume

$$\sigma_{\text{LHS}}^2 \approx \frac{\sigma_{>1}^2}{n} \quad \text{and} \quad \sigma_{\text{MC}}^2 \approx \frac{\sigma^2}{n}. \quad (2.99)$$

Then, using (2.97) we get

$$\frac{\sigma_1^2}{\sigma^2} = 1 - \frac{\sigma_{>1}^2}{\sigma^2} \approx 1 - \frac{\sigma_{\text{LHS}}^2}{\sigma_{\text{MC}}^2}. \quad (2.100)$$

We can estimate σ_{LHS} and σ_{MC} by taking the sample variance across the estimates from several LHS and Monte Carlo runs, respectively. This gives a way of using LHS to estimate the contribution of one dimensional ANOVA terms to the variance of f . We use this estimate in Sect. 2.6 to study the efficiency of LHS for different examples, and illustrate the conditions when Sobol' points perform better than LHS. For now we discuss these conditions theoretically.

The numerical results in [WS07] indicate why QMC, and Sobol' points in specific, can outperform LHS. We enumerate three types of functions for which Sobol' points can provide quadrature with improved integration errors, along with the corresponding features of Sobol' points that enable the improvement.

- 1) *High contribution from one dimensional ANOVA components:* This is the class of functions for which LHS provides large improvements over standard Monte Carlo. Numerical results in [WS07] show that

the discrepancies of one dimensional projections of Sobol' points are even better than for LHS. This allows us to retain the advantages that LHS provides: low variation in integration of the one dimensional parts of the integrand f .

- 2) *High contribution from one dimensional ANOVA components, but truncation dimension $s_T > 1$:* LHS is not able exploit small truncation dimension if it is greater than 1, since it only targets one dimensional components of the integrand. If we take any set $\mathbf{u} = \{1, \dots, l\}$, $1 < l \ll s$ of the early dimensions of an LHS sample, the corresponding discrepancy can be as high as that for pseudorandom point sets. However, for l around 10 or less, the discrepancy of the early dimensions of a Sobol' point set can be much lower for practical sample sizes. Hence, for integrands with truncation dimension $s_T \leq 10$, Sobol' points may provide significant improvement in quadrature error, compared to LHS and random sampling.
- 3) *High contribution from higher dimensional ANOVA components with small truncation dimension $s_T > 1$:* This condition on f further expands the class of functions from item 2, since now we allow higher dimensional ANOVA components to have a large contribution to the function variance, as long as the corresponding dimensions are from the early dimensions of the point set. Clearly, LHS provides no extra advantage beyond that for the one dimensional projections. Sobol' points, however, do. The discrepancy of the projection of Sobol' points onto some subset \mathbf{u} , with small $|\mathbf{u}| > 1$, tends to be lower than that for LHS, as long as the subset is from the early dimensions; i.e., $\mathbf{u} \subset \{1, \dots, l\}$ for small $l > 1$. These conditions on f are significantly less restrictive in practice than those for LHS quadrature being the best option, and suggest that Sobol' sequences – and any other competitive LDS – will perform better than, or as well as, LHS in general.

The reader is referred to [WS07] for some convenient mathematical constructs for the discrepancy of projections of any point set, and similar discussions using these constructs. We stress here that all the theoretical results presented here to illustrate the implications of low effective dimension for QMC are suggestive since they rely on bounds and asymptotes (e.g., the Koksma–Hlawka bound) and not exact relations. There may be cases where QMC performs well even with high effective dimension, as shown in [Tez05] for a class of functions that have full effective dimension in both the truncation and superposition senses. Tezuka shows that for these functions the QMC error decreases as $O(n^{-1})$, without the troublesome $\log^s(n)$ in the numerator, and that the

Koksma–Hlawka bound is so loose for this case as to be completely useless. This shows that low effective dimension is not necessary for QMC to beat Monte Carlo. Owen [Owe03b] points out that is also not a sufficient condition. Given these caveats however, we and several researchers [MC96][CMO97][ABG98][WF03][Owe03b] believe that low effective dimensions play a significant role in creating the conditions for improved quadrature using QMC as compared to Monte Carlo. We provide some simple examples to illustrate and support these arguments in Sect. 2.6. For now, we believe these suggestive theoretical arguments and the cited references, and propose a flow for applying QMC to statistical analysis of circuits.

2.5 Quasi-Monte Carlo for Circuits

The foregoing sections provide us sufficient information to propose a flow for applying QMC to statistical analysis of circuits. As suggested by discussions in Sect. 2.3.2, we use the Sobol’ sequence as our representative LDS in the proposed flow. Once the construction of the promising Niederreiter–Xing sequences [NX96] becomes feasible, we can use them instead of the Sobol’ sequence in the reasonable hope of even better performance.

2.5.1 The Proposed Flow

From Sect. 2.4.1 we know the importance of using transforms like principal components analysis to maximize the amount of variance in the inputs to the minimum number of early dimensions. Also, PCA is popularly used by researchers and practitioners in EDA [Ism93][CS05] to reduce the number of statistical parameters into a small uncorrelated set while still accounting for most of the variance of the original parameters. Hence, we assume that our QMC flow starts with post-PCA statistical parameters: this enables us to focus on aspects that are truly novel in the context of circuit analysis. In fact, if we have transformed the input sampling space to be the unit cube, then we have effectively used some orthogonal transformation like PCA to obtain independent inputs with the same variance. Another way of exploiting low effective dimension in this setting is to measure the contribution of each input of f to the variation in f , and sort the inputs in decreasing order of this measure. Such a rearrangement of the inputs helps minimize the effective dimension s_T in the truncation sense and exploit the good uniformity of the early dimensions of Sobol’ points, as discussed in Sect. 2.4.2. We refer to such a rearrangement as a *variable-dimension mapping*. The impact of any input on the function can be estimated with some measure

Algorithm 2.2 QMC for statistical simulation of circuits

Require: circuit performance functions $\mathbf{f} = \{f_1, \dots, f_{s_y}\}$, joint probability distribution of inputs $\Pi(\mathbf{x})$, input dimensionality s , and sample size n

- 1: $\pi \leftarrow \text{InputOrdering}(s, \mathbf{f}, \Pi) - \pi(j) \in \{1, \dots, s\}$ is the j -th most important input index
 - 2: skip $2^{\lceil \log_2 n \rceil}$ points of the s dimensional Sobol' sequence
 - 3: **for** $i = 1$ to n **do**
 - 4: $\mathbf{z} \leftarrow \text{NextSobolPoint}()$
 - 5: $x_{\pi(j)} = z_j, j = \{1, \dots, s\}$
 - 6: $\mathbf{x}_i = \{x_1, \dots, x_s\}$
 - 7: evaluate $\mathbf{y}_i = \mathbf{f}(\Pi^{-1}(\mathbf{x}_i))$
 - 8: **end for**
 - 9: **return** QMC sample points $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$
-

of global sensitivity, as in [SK05] or in Sect. 1.6.3.1 of this thesis. Here we use one of two much simpler options:

- 1) The designer can select the parameters that most affect the relevant performance metrics, and these can be assigned to the initial dimensions of the QMC. This can be a feasible option in manual design settings where the statistical parameters correspond to different devices in the circuit being designed, since circuit designers often have good insight regarding the devices that significantly affect the relevant performance metrics.
- 2) Run a small standard Monte Carlo run and compute Spearman's rank correlation coefficient between each input x_i and the circuit performance metric. Use this rank correlation as the measure of global sensitivity and sort the inputs in decreasing order of correlation before running QMC. For multiple performance metrics, use the sum of the rank correlation coefficients across all metrics with each input. Spearman's rank correlation is more robust than Pearson's linear correlation in the presence of nonlinear relationships. For an explanation of Spearman's rank correlation, please refer to Sect. 1.6.4.1. Of course, better sampling techniques, like Latin hypercube sampling, or more accurate estimates of global sensitivity, if available, may be used here. However, this simple approach also proves to be sufficiently useful, as demonstrated by the experimental results in Sect. 2.6.

Our proposed QMC algorithm is shown as Algorithm 2.2. The function `InputOrdering()`, shown as Algorithm 2.3, performs the global sensitivity computation to determine a permutation π such that $\pi(j)$ gives

Algorithm 2.3 The function `InputOrdering()` used in Algorithm 2.2

Require: circuit performance functions $\mathbf{f} = \{f_1, \dots, f_{s_y}\}$, joint probability distribution of inputs $\Pi(\mathbf{x})$, input dimensionality s

```

1:  $\rho_i = 0, i = 1, \dots, s$ 
2: for  $i = 1$  to  $n$  do
3:   randomly generate  $\mathbf{x}_i = (x_{i1}, \dots, x_{is})$  from  $\Pi$ 
4:   evaluate  $\mathbf{y}_i = (y_{i1}, \dots, y_{is_y})$  using  $y_{ij} = f_j(\mathbf{x}_i)$  for  $j = \{1, \dots, s_y\}$ 
5: end for
6: for  $j = 1$  to  $s_y$  do
7:   for  $k = 1$  to  $s$  do
8:      $\rho_k = \rho_k + |\text{RankCorr}(\{x_{ik}\}_{i=1}^n, \{y_{ij}\}_{i=1}^n)|$ 
9:   end for
10: end for
11: return  $\pi : \{1, \dots, s\} \rightarrow \{1, \dots, s\}$  such that  $\rho_{\pi(j)}$  is the  $j$ -th largest element in  $\{\rho_k\}_{k=1}^s$ 

```

the index of the input with the j -th largest measure of global sensitivity. In our implementation, this computation involves a $n_\rho = 1,000$ -point Monte Carlo run followed by computation of the rank correlation coefficients. Note that, in Algorithm 2.2, we skip the first $2^{\lfloor \log_2 n \rfloor}$ points of the Sobol' sequence, as recommended empirically in [ABG98] for better performance. The function `NextSobolPoint()` uses the smallest degree primitive polynomials and direction numbers satisfying Sobol's Property A, as discussed in Sect. 2.3.3. The function `RankCorr()` in Algorithm 2.3 computes Spearman's rank correlation, as per Sect. 1.6.4.1.

The sample points returned by QMC can be used for computing some metric, like circuit yield (Sect. 2.2.1.2) or the 99-th percentile, or for further analysis, like visualization or response surface modeling.

2.5.2 Estimating Integration Error

In practice, the exact value of $Q = \int f(\mathbf{x})d\mathbf{x}$ is unknown, for example the exact value of circuit yield. Usually, this is the reason for using numerical quadrature methods. Then how do we estimate the error in the quadrature estimate Q_n ? Random methods, namely Monte Carlo, make this easy since the variance of the Monte Carlo estimate can be used as a probabilistic measure of the error.

2.5.2.1 Estimating Monte Carlo Error

Theorem 2.3 in Sect. 2.2.2 shows us how, using the central limit theorem, we can approximate the distribution of the Monte Carlo error as being

normal, and derive such a probabilistic measure of error. In practice, relying on this assumption of normality, we can use the sample standard deviation of the estimates from several different n -point Monte Carlo runs, to compute this probabilistic measure. Suppose we computed n_{MC} estimates $\{Q_n^{(i)}\}_{i=1}^{n_{\text{MC}}}$. The sample standard deviation is then given by

$$\hat{\sigma}_{\text{MC}}^2 = \frac{\sum_{i=1}^{n_{\text{MC}}} (Q_n^{(i)} - \bar{Q}_n)^2}{n_{\text{MC}} - 1}, \quad (2.101)$$

where the sample mean \bar{Q}_n is given by

$$\bar{Q}_n = \frac{\sum_{i=1}^{n_{\text{MC}}} Q_n^{(i)}}{n}. \quad (2.102)$$

Then the *magnitude* of the Monte Carlo error is within

$$\hat{\sigma}_{\text{MC}} \Phi^{-1} \left(\frac{1+p}{2} \right) \quad (2.103)$$

with probability p , where Φ is the standard normal cumulative distribution function. This corresponds to the *confidence interval* with a confidence level of p .

2.5.2.2 Estimating QMC Error with Scrambled Sequences

Quasi-Monte Carlo is a deterministic quadrature technique: we get the same estimate Q_n every time we run QMC with the same number of points, assuming no changes in the parameters of the LDS (e.g., primitive polynomials for Sobol' points). Hence, there is no natural variance that we can exploit to estimate the error as in the case of Monte Carlo. Also, bounds on the error, like the Koksma–Hlawka bound (2.26), do not help because of at least two reasons:

- It is usually computationally infeasible to estimate both $V(f)$ and D_n^* with acceptable accuracy. It should be noted here that some versions of discrepancy can be computed in reasonable time. Warnock [War72] derived an explicit formula for the L_2 star discrepancy, that was generalized in [CMO97]. See [Hic98] for some generalized error bounds. However, computing the variation of the function still remains infeasible.
- Even if the error bound can be computed, it can be very different from the actual error value, as discussed in Sects. 2.3.1 and 2.3.2.3.

One way to get around this problem is to artificially randomize the QMC points. Then, we can run randomized QMC several times and estimate probabilistic error values, just as we did for Monte Carlo. Several

schemes for randomizing deterministic LDSs have been proposed, and are surveyed in [LL02]. Owen [Owe95] proposed a randomization scheme that *scrambles* (t, s) -sequences and (t, m, s) -nets while maintaining two important properties:

- 1) Every point in the scrambled set has a uniform distribution over C^s , so that the approximation Q_n is unbiased.
- 2) The resulting nets or sequences, are still (t, m, s) -nets and (t, s) -sequences in base, respectively, b with probability one, and with no change to t , m or b .

As mentioned in Sect. 2.3.4, a Latin hypercube sample falls under this class of scrambled (t, m, s) -nets. Since scrambled sequences have the two properties mentioned above, they obey the asymptotic properties and error bounds of their deterministic counterpart. Hence, we can use multiple runs with different scramblings to estimate the variance σ_{QMC} of randomized QMC and the corresponding probabilistic error estimates using (2.103), with σ_{MC} replaced by σ_{QMC} .

For theoretical results on the variance of randomized (t, m, s) -nets, see [Owe97a][Owe97b][Owe98b] by Owen. Owen shows that under certain smoothness conditions on the integrand, scrambled (t, m, s) -nets can actually achieve variance of

$$O\left(\frac{\log^{s-1}(n)}{n^3}\right), \quad (2.104)$$

implying an asymptotic integration error rate of $n^{-1.5}$, which is even better than the standard QMC asymptotic error rate. The smoothness condition requires that the mixed partial derivative,

$$h(\mathbf{x}) = \frac{\partial^s f}{\partial x_1 \dots \partial x_s}, \quad (2.105)$$

satisfies the Lipschitz condition,

$$|h(\mathbf{x}) - h(\mathbf{x}')| \leq B \|\mathbf{x} - \mathbf{x}'\|_2^\beta, \quad (2.106)$$

for some finite $B \geq 0$ and $\beta \in (0, 1]$. Although we will see an example of this rate in the results section (Sect. 2.6), these properties of scrambled nets and their implications for statistical circuit analysis, are not studied in detail in this thesis and can be a fruitful target for future research.

2.5.3 Scrambled Digital (t, m, s) -Nets and (t, s) -Sequences

2.5.3.1 Owen's Scrambling

Let $\{\mathbf{x}_1, \mathbf{x}_2, \dots\}$ and $\{\mathbf{z}_1, \mathbf{z}_2, \dots\}$ denote the original sequence and a randomly scrambled version, respectively, both in base b . Let $x_n^{(i)}$ be the i -th coordinate of \mathbf{x}_n , and let its b -ary expansion be

$$x_n^{(i)} = \sum_{j=1}^{\infty} x_{i,j} b^{-j} = 0.x_{i,1}x_{i,2}\dots, \quad i = \{1, \dots, s\}, \quad (2.107)$$

where $x_{i,j} \in \{0, \dots, b-1\}$ is a digit in base b . Note that we have dropped the subscript n in the expansion, to reduce notation clutter. Assume similar meanings for $z_n^{(i)}$ and $z_{i,j}$. Then,

$$z_{i,1} = \pi^i(x_{i,1}), \quad (2.108)$$

where $\pi^i : \{0, \dots, b-1\} \rightarrow \{0, \dots, b-1\}$ is a randomly chosen permutation for dimension i ; a different such permutation is chosen randomly for each dimension. This operation is, thus, scrambling the first digit of every coordinate. Similarly, we scrambling all other digits with independent, randomly chosen permutation schemes. Furthermore, the permutation of the j -th digit depends on the precise values of the previous $j-1$ digits. We write this as

$$z_{i,j} = \pi_{x_{i,1}, x_{i,2}, \dots, x_{i,j-1}}^i(x_{i,j}). \quad (2.109)$$

For example, the permutation scheme of the third bit in 0.111 will be separate from the permutation scheme of the third bit in 0.101 even though the value of the third bit is the same in both cases. This is because the entire sequence of bits before the third bit determines the permutation applied to the third bit.

Such a scrambling scheme can be computationally tedious because of the extensive bookkeeping required: the number of permutations is $s \frac{b^m - 1}{b - 1}$ for m digits. Less expensive scrambling schemes have been proposed in [Mat98][FT02][Owe03a], among others. Owen [Owe03a] also studies the variance of quadrature estimates from these different scrambling schemes. We use the *linear matrix scrambling* method for digital nets and sequences, as implemented in [HH03].

2.5.3.2 Linear Matrix Scrambling: A Simpler Scheme

Let us rewrite the digital construction of (2.52) in current notation. For integer $n = 1, 2, \dots$, the b -ary expansion of $n - 1$ is

$$n - 1 = \sum_{k=0}^{\infty} a_k b^k = \dots a_2 a_1 a_0. \quad (2.110)$$

Then the digital construction of the i -th coordinate of n -th point is

$$\begin{pmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \end{pmatrix} = \mathbf{C}^{(i)} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \end{pmatrix} \pmod{b}, \quad (2.111)$$

where we have suppressed n to reduce clutter. The linear matrix scrambling method modifies this construction as

$$\begin{aligned} \begin{pmatrix} z_{i,1} \\ z_{i,2} \\ \vdots \end{pmatrix} &= \mathbf{L}^{(i)} + \mathbf{e}^{(i)} \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \end{pmatrix} \\ &= \mathbf{L}^{(i)} \mathbf{C}^{(i)} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \end{pmatrix} + \mathbf{e}^{(i)} \pmod{b}, \quad i = 1, \dots, s. \end{aligned} \quad (2.112)$$

Here, $\mathbf{L}^{(i)}$ are randomly and independently chosen non-singular lower-triangular matrices over $Z_b = \{0, \dots, b - 1\}$, of size $\infty \times \infty$ for infinite precision, and $m \times m$ for m digits of precision. $\mathbf{e}^{(i)}$ are randomly and independently chosen vectors over Z_b , of length same as $\mathbf{L}^{(i)}$. The n -th scrambled point is

$$\mathbf{z}_n = (0.z_{1,1}z_{1,2}\dots, 0.z_{2,1}z_{2,2}\dots, \dots, 0.z_{s,1}z_{s,2}\dots). \quad (2.113)$$

This scrambling method is clearly much simpler than Owen's scrambling, but is also less rich in its range of random permutations. For example, from (2.112), the first digit $z_{i,1} = l_{11}^{(i)} x_{i,1} + e_1^{(i)}$, where $l_{11}^{(i)}$ and $e_1^{(i)}$ are the first elements of $\mathbf{L}^{(i)}$ and $\mathbf{e}^{(i)}$, respectively. $l_{11}^{(i)} \in \{1, \dots, b - 1\}$ for non-singular $\mathbf{L}^{(i)}$, and $e_1^{(i)} \in \{0, \dots, b - 1\}$, giving us $b(b - 1)$ possible permutations, while in Owen's method we have $b!$ possibilities. However, this smaller range of possibilities is not too restrictive and suffices for our experiments. We now show how this scrambling technique can be incorporated into Sobol's construction using direction numbers.

2.5.3.3 Scrambling Sobol' Sequences with Linear Matrix Scrambling

We know from (2.59) that the j -th column of $\mathbf{C}^{(i)}$ contains the bits of j -th direction number $v_j^{(i)}$ for dimension i . Let $\mathbf{v}_j^{(i)}$ denote the vector of the bits of $v_j^{(i)}$; i.e., $\mathbf{v}_j^{(i)}$ is the j -th column of $\mathbf{C}^{(i)}$. If we write

$$\mathbf{L}^{(i)}\mathbf{C}^{(i)} = \mathbf{L}^{(i)}[\mathbf{v}_1^{(i)} \ \mathbf{v}_2^{(i)} \ \dots] = [\mathbf{v}'_1^{(i)} \ \mathbf{v}'_2^{(i)} \ \dots], \quad (2.114)$$

then, denoting $l_{jk}^{(i)}$ as the (j, k) -th element of $\mathbf{L}^{(i)}$, and $v_{j,k}^{(i)}$ as the k -th bit in $\mathbf{v}_j^{(i)}$, we get

$$\mathbf{v}'_j^{(i)} = \begin{pmatrix} l_{11}^{(i)}v_{j,1}^{(i)} + l_{12}^{(i)}v_{j,2}^{(i)} \dots \\ l_{21}^{(i)}v_{j,1}^{(i)} + l_{22}^{(i)}v_{j,2}^{(i)} \dots \\ \vdots \end{pmatrix}. \quad (2.115)$$

Let $\mathbf{l}_j^{(i)} = (l_{1j}^{(i)}, l_{2j}^{(i)}, \dots)^T$ be the j -th column of $\mathbf{L}^{(i)}$. Then, using bitwise Boolean operations we can write

$$\mathbf{v}'_j^{(i)} = v_{j,1}^{(i)} \cdot \mathbf{l}_1^{(i)} \oplus v_{j,2}^{(i)} \cdot \mathbf{l}_2^{(i)} \oplus \dots, \quad 1 \leq i \leq s, \ j > 0. \quad (2.116)$$

This bit vector, $\mathbf{v}'_j^{(i)}$, is the j -th column of the i -th *scrambled* generator matrix given by $\mathbf{L}^{(i)}\mathbf{C}^{(i)}$, and corresponds to a new scrambled direction vector $v'_j^{(i)}$ for the Sobol' construction. Then, corresponding to (2.60), the i -th coordinate for the n -th point is given by

$$z_n^{(i)} = a_0 v'_1^{(i)} \oplus a_1 v'_2^{(i)} \oplus \dots \oplus a_{m-1} v'_m^{(i)} \oplus \mathbf{e}^{(i)}. \quad (2.117)$$

If we use the Gray code construction as in (2.79), we need to XOR $\mathbf{e}^{(i)}$ only once, to the first point, and subsequent points are given simply as

$$z_{n+1}^{(i)} = z_n^{(i)} \oplus v_l'^{(i)}, \quad (2.118)$$

where l is the index of the bit where the Gray codes of $n-1$ and n differ. We are now well-equipped to demonstrate the performance of QMC using experiments. We do this in the next section.

2.6 Experimental Results

In Sect. 2.4.2 we concluded, based on theoretical considerations, that QMC using Sobol' points should result in smaller errors and possibly faster convergence, when compared to Latin hypercube sampling. We

now test this conclusion experimentally on some simple examples that also allow us to validate the results analytically. After this, we demonstrate the performance of QMC on a variety of circuit benchmarks, in comparison with standard Monte Carlo and LHS.

2.6.1 Comparing LHS and QMC (Sobol' Points)

Let f be our integrand. We test two conclusions from Sect. 2.4.2 here:

- 1) LHS almost completely and exclusively removes the variance contribution of the one dimensional ANOVA components of f , to achieve variance reduction. Hence, we can estimate the variance contribution of the one dimensional components via (2.100). See Sect. 2.4.1 for a discussion on the ANOVA decomposition.
- 2) LHS restricts its variance reduction activity to the one dimensional components of f . Sobol' points provide further benefit by reducing the error in integrating also some higher dimensional components, because they enjoy highly uniform higher dimensional projections in the early dimensions.

2.6.1.1 LHS (Almost) Exactly Removes One Dimensional Variance Contribution

Consider the following three functions in five dimensions.

- 1) *An additive function* with only one dimensional nonzero components; i.e., with effective dimension s_S equal to 1, in the superposition sense.

$$f_a = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2. \quad (2.119)$$

Note that the function has full truncation dimension $s_T = 5$. We expect LHS to almost completely remove any variance in the integral estimate for this function, because there is no contribution from any multi-dimensional components.

- 2) *A cross-term function* with significant contributions from multi-(two-)dimensional components; i.e., with superposition dimension $s_S = 2$. The truncation dimension is still 5.

$$f_c = (x_1 + x_2 + x_3 + x_4 + x_5)^2. \quad (2.120)$$

Note that f_a is part of f_c . We expect the effectiveness of LHS to be less for this function, and that the remaining variance in the estimate (σ_{LHS}^2) is proportional to the variance contribution from the two dimensional components.

	f_a	f_c	f_s
$\hat{\sigma}_{\text{MC}}^2$	4.024×10^{-5}	9.483×10^{-4}	6.112×10^{-4}
$\hat{\sigma}_{\text{LHS}}^2$	5.284×10^{-13}	2.839×10^{-5}	2.839×10^{-5}
$\hat{\eta}_{\text{LHS}} = \frac{\hat{\sigma}_1^2}{\hat{\sigma}^2} = 1 - \frac{\hat{\sigma}_{\text{LHS}}^2}{\hat{\sigma}_{\text{MC}}^2}$	1.000	0.970	0.954
Exact σ^2	—	$193/18$	$125/18$
Exact σ_1^2	—	$94/9$	$20/3$
$\eta = \frac{\sigma_1^2}{\sigma^2}$	1	0.974	0.960

Table 2.3. Fractional variance contribution from one dimensional components of f_a , f_c and f_s computed using LHS estimate (2.100) and analytically. We see that LHS does exclusively remove the variance from one dimensional components

- 3) A *strongly cross-term function* with even higher relative contribution from two dimensional components.

$$f_s = f_c - f_a = 2 \sum_{i=1}^4 \sum_{j=i+1}^5 x_i x_j. \quad (2.121)$$

We expect that the variance of LHS, σ_{LHS}^2 will not change from the case of f_c since we have only removed the one dimensional components and not changed anything in the two dimensional components.

We ran 30 Monte Carlo runs and 30 LHS runs, each with a sample size of $n = 10,000$, to estimate the Monte Carlo variance (σ_{MC}^2) and the LHS variance (σ_{LHS}^2), respectively. We use the sample variance formula (2.101). Plugging these variance estimates into (2.100), we can estimate the fraction of variance contributed by the one dimensional components of f . Let σ^2 be the total function variance, σ_1^2 the variance from the one dimensional components and $\eta = \sigma_1^2/\sigma^2$ the fraction of variance from one dimensional components. Table 2.3 shows the results in data rows 1–3. We can see that, as expected, the LHS variance for the additive function f_a is negligibly small. Since f_a has only one dimensional components, all its variance is from one dimensional components. The estimate $\hat{\eta}_{\text{LHS}}$ is almost exact ($= 1$). For the other functions, we can analytically compute σ^2 , σ_1^2 and η . These exact values are shown in data rows 4–6. Derivations are given in Appendix A. We see that the estimates $\hat{\eta}_{\text{LHS}}$ are very close to the exact values, providing strong evidence for the claim that LHS exclusively removes σ_1^2 from the estimate variance.

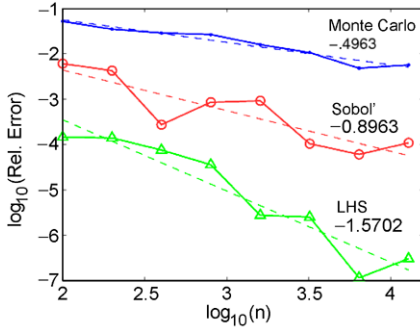
2.6.1.2 Sobol' Points Are Better Than LHS for Functions with Significant Higher Dimensional Components

All three test functions above allow a simple analytical computation of their exact integrals over the unit cube C^s , this being one of the reasons for choosing them as test functions. These exact values are $Q(f_a) = 5/3$, $Q(f_c) = 20/3$ and $Q(f_s) = 5$. We also computed these integrals numerically in three different ways:

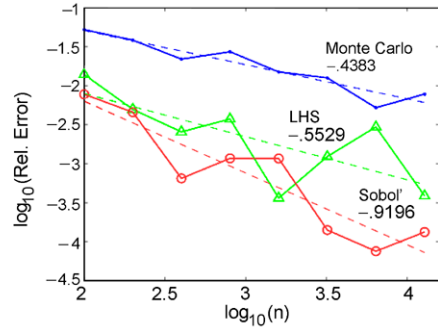
- 1) Using standard Monte Carlo with increasing number of points n . The values of n are chosen to match those for the LHS samples sizes below.
- 2) Using Latin hypercube samples with sample sizes of $n = 100 \cdot 2^{\{0, \dots, 7\}} = \{100, 200, 400, \dots, 12,800\}$.
- 3) Using Sobol' point with the same samples sizes as LHS.

Since we know the exact answers, we can directly compute the relative error without having to resort to probabilistic errors based on sample variance. The plots in Fig. 2.8 show the relative integration error for these three methods on all three test functions on a $\log_{10} - \log_{10}$ scale. Least squared-error linear fits in this scale, shown as dashed straight lines, estimate the convergence exponent of each integration method as the slope of the fit. These estimated rates are annotated on the corresponding linear fits. We now discuss each of the three test cases in some detail, in the context of these results.

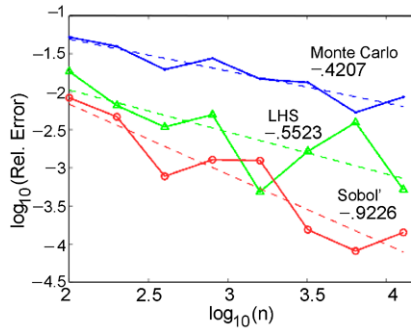
- *Additive function f_a (Fig. 2.8(a))*: Monte Carlo achieves an estimated error rate of $n^{-0.4963}$, which is close to the expected $n^{-0.5}$ rate. The Sobol' points achieve a rate of $n^{-0.8963}$, which is close to the asymptotic rate of n^{-1} for QMC. This suggests that the truncation and superposition dimensions ($s_T = 5, s_S = 1$) are small enough for the Sobol' points to exploit. Interestingly, LHS achieves a *much* faster convergence of $n^{-1.5702}$ along with lower error. This superiority of LHS over Sobol' points is actually expected. We know, from Sect. 2.3.4, that an LHS sample is a scrambled (t, m, s) -net. Since the function f_a satisfies Owen's smoothness condition (2.106), the result in (2.104) predicts an asymptotic $n^{-1.5}$ convergence for LHS error. Similar behavior is predicted by Fox in [Fox99], Theorem 9.1.2.
- *Cross-term function f_c (Fig. 2.8(b))*: Even with contribution from higher dimensional components in f , Monte Carlo achieves more or less the same convergence rate ($n^{-0.4383}$), close to the expected, asymptotic $n^{-0.5}$. However, we see a big change in the performance



(a) Results for additive function
 $f_a = \sum_{i=1}^5 x_i^2$



(b) Results for function with
 cross terms, $f_c = (\sum_{i=1}^5 x_i)^2$



(c) Results for function with relatively
 stronger cross terms, $f_s = f_c - f_a$

Figure 2.8. Comparison of relative errors of Monte Carlo, LHS and QMC (Sobol' points) with increasing number of points. The three test functions have different relative contributions from their one dimensional ANOVA components

of LHS. It provides no benefit over Monte Carlo in integrating the *two* dimensional components of f , resulting in an overall error rate of $n^{-0.5529}$, which is closer to the Monte Carlo error rate. Note that it is still significantly better than Monte Carlo because its excellent performance on the one dimensional components. Interestingly, the Sobol' points maintain their low error and fast convergence in spite of the increased superposition dimension of the integrand. This supports our argument that Sobol' points have good uniformity in higher dimensional projections of the early dimensions, which allows them to integrate the higher dimensional ANOVA components with better accuracy than Monte Carlo or LHS.

- *Strong cross-term function f_s (Fig. 2.8(c))*: This function has even higher relative contribution from its two dimensional components. However, we see no significant difference in the performance of any

of the three methods. This is not surprising. The variance of Monte Carlo, of course, does not exploit any ANOVA features of the integrand. This explains the lack of change in its performance. LHS primarily targets the one dimensional components and has Monte Carlo-type performance on higher dimensional components of f_s . f_c and f_s differ only in their one dimensional components and have identical two dimensional components. Hence, the error in the LHS estimate, which is almost completely due to the latter, does not change. Similar arguments apply to the Sobol' points: the change in the error due to changes in the one dimensional components is very small.

Based on these experiments and the arguments in Sect. 2.4.2, we can confidently conclude that, in the general case, Sobol' sequences will show lower error and faster convergence than LHS or Monte Carlo, as long as the truncation dimension of the integrand is not too large. The only exception is when almost all of the variance contribution is due to the one dimensional components of the integrand and the integrand is smooth, in which case LHS or scrambled QMC will perform better. For the case of statistical circuit analysis, we neither expect the integrand to be primarily one dimensional, nor to be smooth (e.g., the characteristic function integrand (2.12) for circuit yield). However, from common design knowledge, we believe that the truncation dimension of the integrand will not be too large. Hence, Sobol' points (or any other competitive QMC method) seem an appropriate choice.

2.6.2 Experiments on Circuit Benchmarks

We now demonstrate the performance of QMC on circuit benchmarks. Before we discuss the benchmarks and the results, we briefly mention some relevant implementation details. A linear congruential generator (LCG) [Gla04] (`drand48()` in C) was used to generate the pseudorandom sequences for standard Monte Carlo and for random scramblings of the Sobol' points. This generator enjoys widespread popularity and the obtained results will be immediately relevant to the general practitioner. Also, variance results in [OE04] comparing LCG with a generalized feedback shift register generator (GFSR) [MK94], do not show significant improvements for GFSR in the context of randomized QMC. The standard Box Muller [BM58] method for generating normally distributed variates is inaccurate, especially for a large number of samples [Tez95]. Hence, an inverse transform method, by Acklam [Ack], was used. This is the Π_i^{-1} in (2.17) for the case of normal variates. Now we describe the benchmark circuits and the experiments. All results will be discussed together after this description. We use the following three benchmarks.

- 1) **Master–slave flip-flop with scan chain (MSFF):** This is the same circuit used as a benchmark for SiLVR in Sect. 1.7.1. In this case, we are computing the parametric yield, given a maximum acceptable clock-output delay (τ_{cq}) of 200 ps. This is a 31 dimensional problem. 10 Monte Carlo runs with 50,000 points each were run to compute the Monte Carlo variance. The QMC variance is computed across a set of 10 runs: one run using 50,000 Sobol' points and 9 runs using 50,000 scrambled Sobol' points each. Each scrambled run uses a distinct set of permutations.
- 2) **Sub-1 V CMOS bandgap voltage reference:** This benchmark is also used for testing SiLVR in Sect. 1.7.3, where a detailed description is also provided. In this case, we compute the parametric yield, given three specifications: 1) output voltage, V_{ref} within 10% of 600 mV, 2) output settling time $\tau_s \leq 200$ ns, and 3) dropout voltage $V_{do} \leq 900$ mV. The settling time is defined as the time taken by the output to settle within 1% of its final value. This is a 122 dimensional problem. We use the same run plan as for the flip-flop benchmark, but with a sample size of 20,000 for each run.
- 3) **64-bit SRAM column:** This benchmark is also used for testing Statistical Blockade in Sect. 3.3.4, where a detailed description is also given. In this case, we are computing the 90-th percentile of the write time τ_w in the presence of manufacturing variations. This is a 403 dimensional problem. The same run plan as for the flip-flop is used, with the only difference being the sample size for each run: here we use 10,000 points.

It is clear that the problem dimensions are large enough such that 10,000–50,000 Sobol' points will not be uniformly distributed over all dimensions. We are bound to get undesirable patterns in several projections, similar to the ones shown in Sect. 2.4. Here, it becomes important that we use some technique to reduce the effective dimension of the problem. As described in Sect. 2.5.1, we use Spearman's rank correlation (1.69) as a measure of variable importance (or global sensitivity), and arrange the statistical parameters in decreasing order of importance, before running QMC.

As an illustrating example, let us look at how the rank correlation based variable-dimension mapping works for the flip-flop, shown in Fig. 2.9(a). Figure 2.9(b) shows the magnitude of the rank correlation ($|\rho_S|$) of each parameter with the clock-output delay for rising output, computed from an initial Monte Carlo run of 1,000 samples. The variables are sorted in *decreasing* order of importance (rank correlation mag-

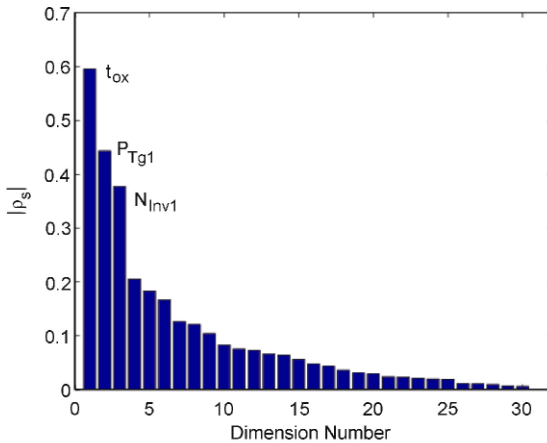
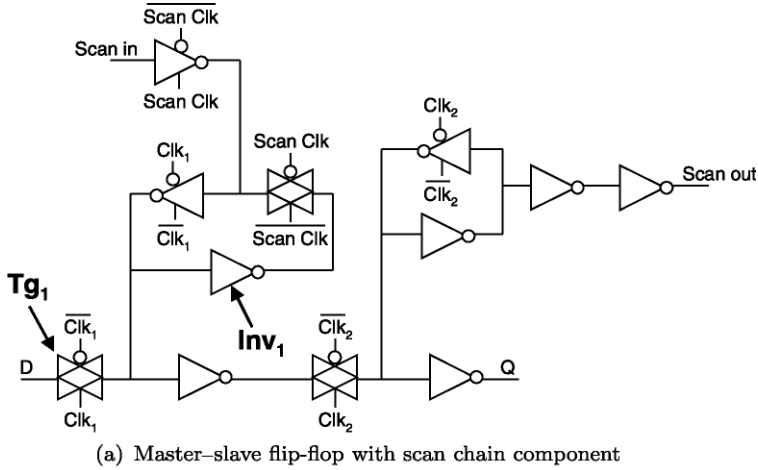


Figure 2.9. These figures illustrate the use of rank correlation as a measure of parameter importance, to be used for variable-dimension mapping

nitude): this is now the order they will be mapped to the increasing dimensions of the Sobol' sequence. The three most important parameters are labeled: 1) t_{ox} : global gate oxide variation, 2) P_{Tg1} : the V_t variation in the pMOS device in the input transmission gate Tg_1 , and 3) N_{Inv1} : the V_t variation in the nMOS device in the inverter Inv_1 . The latter two devices are on the critical signal path for a high input causing a rising output, and are important for correctly sampling a "1" at the input, especially when the input timing is close to the setup limit. Since the input was timed in such a manner in the testbench, these measures of importance make intuitive sense.

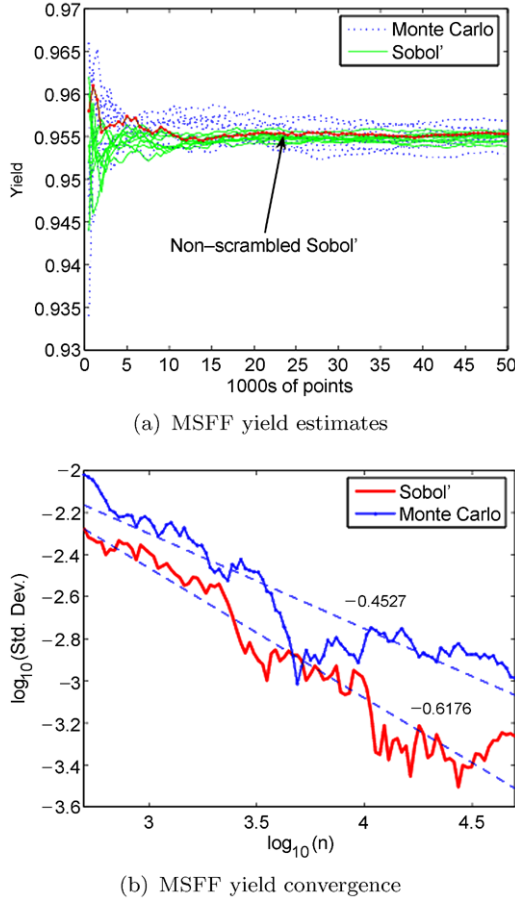
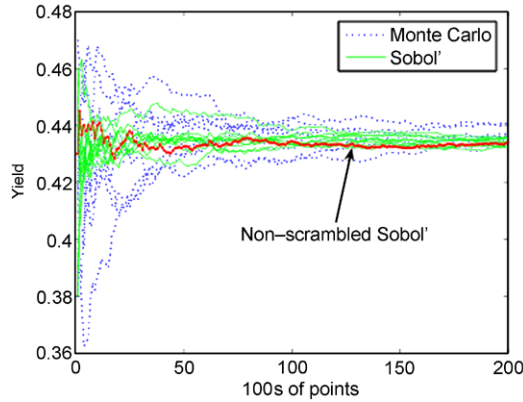


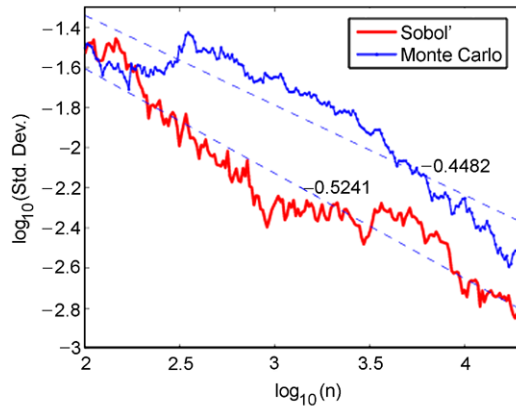
Figure 2.10. Comparison of Monte Carlo and QMC estimates and std. deviation convergence, for the flip-flop benchmark

2.6.2.1 Analysis of Results

Figures 2.10(a), 2.11(a) and 2.12(a) plot the values of the estimates with increasing number of points for each Monte Carlo (pseudorandom) and QMC (Sobol') run. For all three cases, we can clearly see that the QMC graphs converge more quickly than the Monte Carlo graphs in general. In particular, the *non-scrambled* Sobol' points converge very fast towards the final result. This fact provides indirect validation that our rank correlation based dimension mapping is an effective heuristic. Scrambling the digits of an LDS sample changes the way the sampling space is filled up, and hence, changes the patterns and the discrepancies of the projections of the point set. We observe here that changing the patterns in this way causes the QMC performance to degrade



(a) Bandgap yield estimates



(b) Bandgap yield convergence

Figure 2.11. Comparison of Monte Carlo and QMC estimates and std. deviation convergence, for the voltage reference benchmark

in general for our benchmarks. This implies that the rank correlation arranges the variables in a way that is optimal (or at least, advantageous), given the patterns of the non-scrambled LDS. This behavior is more pronounced as the problem dimensionality increases from MSFF to the SRAM Column, suggesting that for low dimensionality (e.g., 31 dimensional MSFF), the LDS uniformity does not show large variation for different projections. For high dimensional problems, however, effective variable-dimension mapping should give notable improvement, over a random or uneducated assignment of variables to LDS dimensions. Of course, the impact of such mappings depends on the minimum possible truncation dimension of the integrand. If the truncation dimension of a problem cannot be made much smaller than the full di-

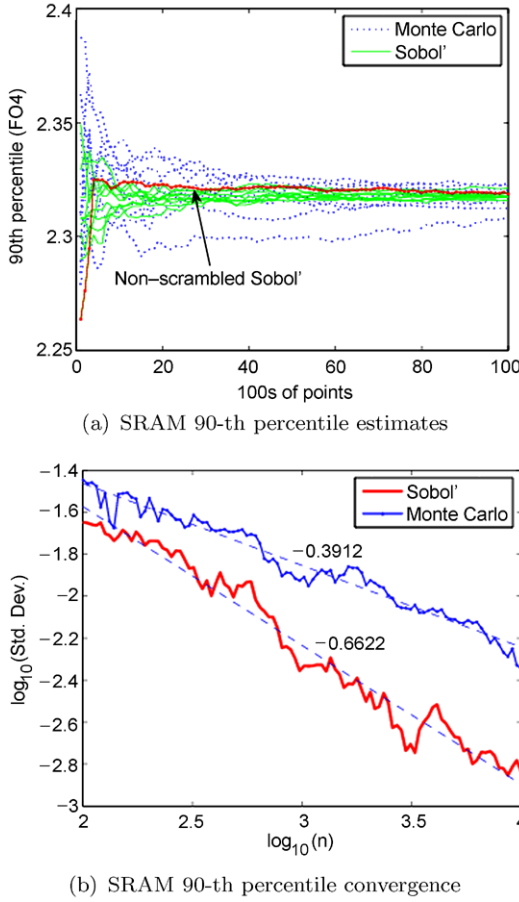


Figure 2.12. Comparison of Monte Carlo and QMC estimates and std. deviation convergence, for the SRAM column benchmark

mensionality, then all mappings will achieve similar performance. Again, in high dimensions, it is very likely that the minimum possible truncation dimension will be much smaller than the full dimensionality. As a result, using the correct mapping will result in much improved performance.

Figures 2.10(b), 2.11(b) and 2.12(b) compare the standard deviation of the Monte Carlo runs ($\hat{\sigma}_{\text{MC}}$) and the QMC runs ($\hat{\sigma}_{\text{QMC}}$) with increasing number of points, showing the effectiveness of scrambled QMC as a variance reduction method. The plots are in $\log_{10} - \log_{10}$ scale, where a $\sigma \propto n^{-\alpha}$ relationship will appear as a straight line with slope $-\alpha$, similar to the plots in Fig. 2.8. Linear fits, via least squared error, are shown as dashed straight lines, and are annotated with the cor-

responding convergence exponent. We can see that, in general, QMC shows lower variance and faster convergence than Monte Carlo across all three benchmarks. The estimated Monte Carlo convergence rates are a little slower than the asymptotic rate of $n^{-0.5}$. This can be because we have not reached the asymptotic rate and also because the estimates are computed from only 10 runs. A larger number of runs is definitely desirable, but eludes us because of the large circuit simulation times. Even with these approximate estimates we do get a good sense of the performance of QMC relative to Monte Carlo. σ_{QMC} shows convergence rates in between the Monte Carlo and QMC asymptotic rates of $n^{-0.5}$ and n^{-1} , respectively. This suggests that the integrands for these benchmarks have superposition dimension greater than 1 and moderately large truncation dimension. Another reason for these reduced rates can be the lack of smoothness in characteristic function integrand (2.12). Integrand smoothness can lead to better QMC performance as indicated in [MC96] and [Fox99]. Figure 2.13 provide evidence for larger than one superposition dimension. Here we plot the standard deviation of LHS estimates for the SRAM column and voltage reference cases. We can see that the LHS curve lies in between the curves for Monte Carlo and QMC. This indicates that there are some significant multi-dimensional components of the integrands for which the Sobol' points further reduce the integration error over LHS, similar to the case of functions f_c and f_s in Fig. 2.8.

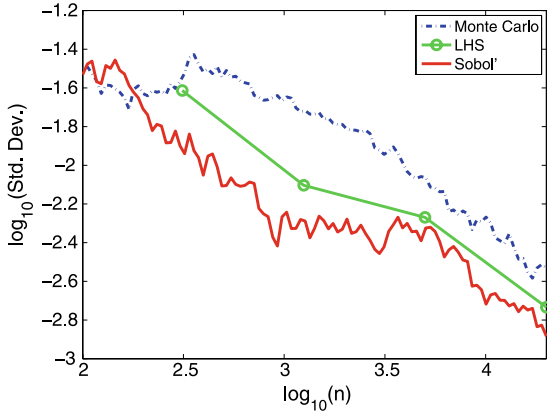
We now compute some estimates of the samples size needed to achieve a given accuracy criterion. Say the exact value of the integral is Q and we specify an accuracy criterion as follows: we want the estimate Q_n to be within $\delta\%$ of Q with a probability of p . In other words, we want the error magnitude to be less than or equal to $Q(\frac{\delta}{100})$. Using the estimate for probabilistic error from (2.103), we can write this as

$$\hat{\sigma}_{\text{MC}}\Phi^{-1}\left(\frac{1+p}{2}\right) \leq Q\left(\frac{\delta}{100}\right). \quad (2.122)$$

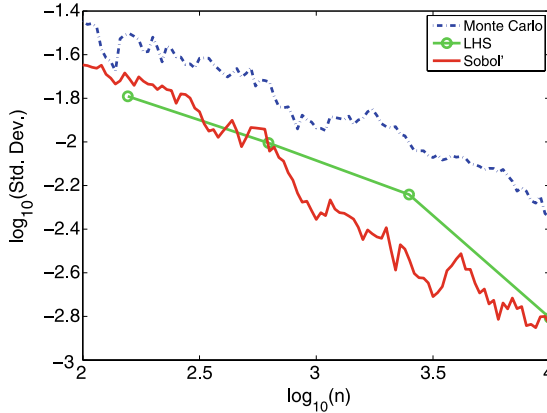
For $p = 0.9545$, we get

$$\hat{\sigma}_{\text{MC}} \leq Q\left(\frac{\delta}{200}\right). \quad (2.123)$$

Using the linear fits from Figs. 2.10(b), 2.11(b) and 2.12(b), we can then estimate the number of points n needed to satisfy this criterion. The same arguments hold for σ_{QMC} and the QMC sample size. The results for $\delta = 1, 0.1$ are shown in Table 2.4. Since, we do not know Q for these circuit benchmarks, we estimate it using *all* the points at our disposal – from 10 Monte Carlo and 10 QMC runs – and assume that the error in this estimate is negligible in comparison with $Q(\frac{\delta}{100})$. Even



(a) Bandgap yield convergence



(b) SRAM 90-th percentile convergence

Figure 2.13. Comparison of std. deviation convergence of Monte Carlo, LHS and QMC

if the error is not negligible, because we are using the same assumption for both Monte Carlo and QMC, the *relative* trends seen here can be believed. We can see moderate to large speedups ($2\times$ to $50\times$), showing the effectiveness of scrambled QMC as a variance reduction method. Furthermore, these speedups tend to improve as the required accuracy increases.

The results presented in this section are promising and recommend using QMC for general circuit analysis problems. Of course, these are initial results and there is much scope for more research in this area. We discuss some immediate directions for future work in the following section.

δ	MSFF	SRAM column	Voltage ref.
	MC / QMC	MC / QMC	MC / QMC
1%	1,114 / 588 (1.9 \times)	1,631 / 354 (4.6 \times)	89,115 / 10,360 (8.6 \times)
0.1%	180,232 / 24,465 (7.4 \times)	586,771 / 11,451 (51.2 \times)	15,182,252 / 838,062 (18.1 \times)

Table 2.4. Number of points needed to achieve a given error with a confidence level of 95.45%. Speedup of QMC over MC is shown in brackets

2.7 Future Work

This study brings up many relevant questions and possibilities. We outline a few here.

- 1) Owen proposes *Latin supercube sampling* (LSS) in [Owe98a]. LSS combines LHS and QMC in an attempt to exploit the excellent properties of QMC for the small dimensional projections, while achieving at least Monte Carlo-type performance for the high dimensional projections. The set of s dimensions is divided into k exclusive subsets. Scrambled QMC is used for each subset, with different scramblings for each subset. Then, the points in each subset are randomly permuted, as in LHS, before combining them together to achieve an s dimensional LSS sample. Each subset enjoys the scrambled QMC rate of convergence and the variance resulting from the interaction between subsets enjoys the Monte Carlo rate. We can see the similarity with LHS, where each subset is of size one. In practice, the number of statistical parameters in a circuit can become extremely large (1,000s or more). In such cases, it will likely be essential to use such mixed sampling methods to achieve effective performance from QMC. Spanier [Spa95] suggests a less powerful, but easier to apply, hybrid sampling technique using QMC for the first d dimensions and Monte Carlo for the rest.
- 2) We saw theoretical results in Sect. 2.5.2 and experiments (on LHS) in Sect. 2.6.1 that suggest that scrambled QMC can achieve up to $n^{-1.5}$ error convergence asymptotically, if the integrand is smooth. Morokoff and Caflisch [MC95] show that the lack of continuity in the integrand can reduce the effectiveness of non-scrambled QMC, resulting in Monte Carlo type performance. Integrands in circuit yield analysis are typically characteristic functions as in (2.12), which are discontinuous at the boundary of the acceptance region where they

suddenly change from 1 to 0. For any arbitrary boundary, whether a QMC point falls within the boundary to contribute a 1 to the integral, or outside the boundary to contribute a 0, is essentially random. This random sampling around the entire boundary leads to the degradation in QMC performance towards Monte Carlo performance. In high dimensions, the boundary becomes relatively more significant (e.g., the ratio of the boundary area of a unit cube to its volume, in s dimensions is $2s$), and the degradation worsens with increasing dimensions. Moskowitz and Caflisch [MC96] shows a method of “smoothing” such integrands by enforcing continuity, without changing the value of the integral. Fox [Fox99] discusses other forms of smoothing in the context of randomized QMC. Using such, or novel, smoothing techniques can help further improve the performance of QMC for circuits with medium dimensionality, and make QMC effective on problems with very large dimensionality.

- 3) Variance reduction techniques [Gla04][Fis06] are widely employed to reduce the variance – and, hence, the error – of standard Monte Carlo. Since QMC shows Monte Carlo type performance on integrands with large effective dimension, variance reduction techniques like control variates, stratification and importance sampling should be very useful in such cases. In fact, LSS is a form of stratified sampling applied to QMC. Some applications of these techniques are discussed in [Fox99].

Novel Algorithms for Fast Statistical Analysis of Scaled
Circuits

Singhee, A.; Rutenbar, R.A.

2009, XV, 195 p., Hardcover

ISBN: 978-90-481-3099-3