JOURNAL OF COMPLEXITY **14**, 527–556 (1998)
ARTICLE NO. CM980489

On the L_2-Discrepancy for Anchored Boxes

Jiří Matoušek*

Department of Applied Mathematics, Charles University, Malostranské nám. 25, 118 00 Prague 1, Czech Republic
E-mail: matousek@kam.mff.cuni.cz

Received November 21, 1997

The L_2-discrepancy for anchored axis-parallel boxes has been used in several recent computational studies, mostly related to numerical integration, as a measure of the quality of uniform distribution of a given point set. We point out that if the number of points is not large enough in terms of the dimension (e.g., fewer than 10^4 points in dimension 30) then nearly the lowest possible L_2-discrepancy is attained by a pathological point set, and hence the L_2-discrepancy may not be very relevant for relatively small sets. Recently, Hickernell obtained a formula for the expected L_2-discrepancy of certain randomized low-discrepancy set constructions introduced by Owen. We note that his formula remains valid also for several modifications of these constructions which admit a very simple and efficient implementation. We also report results of computational experiments with various constructions of low-discrepancy sets. Finally, we present a fairly precise formula for the performance of a recent algorithm due to Heinrich for computing the L_2-discrepancy. © 1998 Academic Press

1. INTRODUCTION

Let P be an n-point subset of the d-dimensional unit cube $[0, 1]^d$. Various types of *discrepancy* of P have been considered as quantitative measures of the nonuniformity of distribution of P. Such investigations have been motivated by theoretical interest (see, e.g., [1, 2]) and also by applications in numerical integration and related problems (see, e.g., [16, 18, 28] for recent surveys). Perhaps the most popular type of discrepancy (and also the oldest) is the *discrepancy for anchored axis-parallel boxes*,

* Research supported by Czech Republic Grant GAČR 0194, by Charles University Grants 193 and 194, and by the Czech-U.S. Science and Technology Research Grant 94, 051. Most of the numerical computations were done on machines of the Supercomputing Center of the Charles University, Prague.

often called the *star-discrepancy* for historical reasons. For a point $x \in [0,1]^d$, let B_x denote the box $\prod_{i=1}^d [0, x_i)$. The discrepancy of P for B_x is defined as the deviation of the volume of B_x from the fraction of points of P lying in B_x, that is,[1]

$$D(P, B_x) = x_1 x_2 \cdots x_d - \frac{1}{n}\,|P \cap B_x|.$$

The star-discrepancy of P is defined as $D^*(P) = \sup_{x \in [0,1]^d} |D(P, B_x)|$. Another often-considered notion is the L_2-average of $D(P, B_x)$ over $x \in [0,1]^d$, i.e.

$$D_2^*(P) = \left(\int_{[0,1]^d} D(P, B_x)^2 \, \mathrm{d}x \right)^{1/2}$$

(here the notation is far from being standardized). Since most of the results in existing literature refer to the *squared* L_2-discrepancy, i.e. to $D_2^*(P)^2$, we will follow this convention too. This means that all numerical values in tables etc. refer to the square of the L_2-discrepancy.

This L_2-discrepancy seems much more tractable than the worst-case discrepancy $D^*(P)$, both theoretically and practically. On the theoretical side, the asymptotic behavior of the function $D_2^*(n) = \inf_P D_2^*(P)$, where the infimum is over all n-point sets $P \subset [0,1]^d$, has been determined for every fixed dimension d: We have

$$\frac{c_d}{n} \log^{(d-1)/2} n \leqslant D_2^*(n) \leqslant \frac{C_d}{n} \log^{(d-1)/2} n \tag{1}$$

for all $n \geqslant 2$, with c_d, C_d depending on d only ([24, 25]; the results are also presented in [1]). In the analogous question for $D^*(\cdot)$, only much less precise bounds are known, despite considerable effort. On the practical side, given P, $D_2^*(P)$ can be easily evaluated by reasonably fast algorithms [31, 9], while computing $D^*(P)$ seems practically intractable for dimensions higher than 2 or 3, say.

Constructions providing sets with a low discrepancy have been studied extensively (see Sections 3 and 4 for an overview of some such constructions). For many purposes, such sets are more uniformly distributed than random point sets and, hence, they provide an improvement over

[1] Here we measure the deviation in units of *volume*, as is usual in papers considering applications of discrepancy for numerical integration and similar methods. In most "pure" discrepancy theory papers, discrepancy is measured in units of *points*, i.e. as our quantity multiplied by n.

Monte-Carlo methods for numerical integration and other important tasks. Although it has been argued (e.g., [30]) that this improvement becomes negligible in dimensions over 15 or 20 for numerical integration of some types of functions, recent successful applications of low-discrepancy sequences in financial mathematics deal with much higher dimensions, like 60 of even a few hundred (e.g., [20, 22]).

Since no efficient algorithms are known for evaluating the worst-case discrepancy, computations of L_2-discrepancy have been often used for assessing the quality of various low-discrepancy sets (see, e.g., [15, 29, 11] for recent studies in this spirit). In Section 2, we show that L_2-discrepancy may be a rather misleading parameter if the dimension is high and the number of points is relatively small. The reason is that for a small number of points, nearly the best possible L_2-discrepancy is attained by a set whose all points are clustered near the corner $(1, 1, ..., 1)$. An alternative discrepancy notion, recently defined by Hickernell [10], which assigns a larger importance (weight) to boxes of larger volume, suffers from this effect in a much smaller extent.

A loosely related recent result is due to Sloan and Woźniakowski [27], who prove that, roughly speaking, if a point set with fewer than 2^d points is used for numerical integration in dimension d, no nontrivial estimate for the error holds in the worst case (see [27] for a precise formulation). Both their result and our result mentioned above support the well-known experience that small sets are generally unsuitable for numerical integration in higher dimension, but otherwise the results go in distinct directions (ours concerns the suitability of L_2-discrepancy for assessing a point set, and here bad things occur also for sets considerably bigger than 2^d, while the result of Sloan and Woźniakowski has a sharp threshold at 2^d).

In Section 3, we recall a randomized method for constructing low-discrepancy point sets (and we also describe a reasonably efficient implementation for producing such sets). The expected squared L_2-discrepancy for this construction has recently been analyzed by Hickernell [10]. We suggest several variants of the randomization, some of them apparently new, whose implementation is much simpler and faster, but for which Hickernell's formula for L_2-discrepancy remains valid. We expect that these methods of generating well-distributed sets might be among the most efficient ones in practice for a number of applications, and at the same time good theoretical bounds are available for them.

In Section 4, we describe results of numerical studies of the L_2-discrepancy for several popular constructions of low-discrepancy sets (Richtmyer, Halton, and Faure sequences and randomized variants of the latter two). As was already mentioned, similar studies have been conducted before, but we cover also methods not yet tested and present the results in a somewhat different manner. For dimensions 15 and above, none of the tested constructions gives a significant improvement in L_2-discrepancy over random point sets in the tested range of set sizes (up to 2^{16} points). The L_2-discrepancy for the Faure and Halton constructions displays a somewhat erratic behavior; Faure's method sometimes gives quite bad values. The methods of introducing randomization into these constructions (discussed in section 0) all appear suitable for stabilizing the behavior and obtaining a good L_2-discrepancy in most cases. Currently, the experiments only involved several constructions that are relatively simple and easy to implement. It would be very interesting to conduct a similar (or more extensive) study for more recent and/or complicated constructions, such as various more complicated (t, m, s)*-nets in base* b described by Niederreiter [18], a very recent method by Niederreiter and Xing [19], or the so-called admissible lattices discussed by Skriganov [26].

Finally, in Section 5, we consider an algorithm due to Heinrich [9] for computing the L_2-discrepancy of a given n-point set. A previous, simpler algorithm of Warnock [31] (used in most of empirical studies of L_2-discrepancy) needs roughly dn^2 arithmetic operations for an n-point set. Heinrich obtained the asymptotic bound $C_d n \log^d n$ for the number of operations of his algorithm, with C_d depending (exponentially) on the dimension, and he noted that according to experiments, the algorithm gives a considerable advantage over the $O(dn^2)$ method only for small d's. The asymptotic bound assuming d fixed is not very relevant for actual computations where n cannot by far be large enough in terms of d; a somewhat more realistic setting is to view the quantities n and 2^d as independent variables, where n is somewhat larger than 2^d but not drastically. We analyze the performance almost precisely: For n of the form 2^k, k an integer, we derive a worst-case upper bound of

$$2n \sum_{i=0}^{d} (d+2-i) \binom{k}{i}$$

for the number of arithmetic operations. Heuristic considerations and experiments show that this usually overestimates the actual number of operations for uniformly distributed point sets by a factor between 1.5 and 2. From this formula, one can infer that for d large, Heinrich's algorithm starts outperforming the straightforward method significantly for n around 2^{2d}.

2. DISCREPANCY FOR RELATIVELY SMALL SETS

Let $\mathrm{vol}(B_x) = x_1 x_2 \cdots x_d$ denote the volume of the box B_x. It is not difficult to calculate that the average value of $\mathrm{vol}(B_x)^2$ over x uniformly distributed in $[0,1]^d$ equals 3^{-d}. Therefore, if P_0 is the n-point multiset consisting of n copies of the point $(1, 1, ..., 1)$, the discrepancy of each box B_x equals its volume, and hence, $D_2^*(P_0)^2 = 3^{-d}$.

As is well-known, the expected value of $D_2^*(P)^2$ for a random n-point set $P \subset [0,1]^d$ is $(1/n)(2^{-d} - 3^{-d})$. If we compare this with $D_2^*(P_0)^2$ for the above pathological set P_0, we see a first warning that something bad happens with the L_2 discrepancy for a small number of points: For $n < (1.5)^d - 1$, $D_2^*(P_0)^2$ is better than the expectation for a random point set, although hardly anyone would call P_0 uniformly distributed in the unit cube.

We are going to argue that for a small n, P_0 even has almost the best possible L_2-discrepancy. The reason is that if the dimension is high and n is relatively small, then a typical box has volume smaller than $(1/2n)$, and hence, its discrepancy is smallest if it contains no points.

To get a quantitative version of this statement, we begin by the well-known observation that for any n-point set P and a box B_x, the number of points of P in B_x is integral and so we have

$$|D(P, B_x)| \geqslant \frac{1}{n}\, \delta(n \operatorname{vol}(B_x)),$$

where $\delta(t)$ denotes the distance of a real number t to the nearest integer. Therefore,

$$D_2^*(n)^2 \geqslant \frac{1}{n^2} \int_{[0,1]^d} \delta(n \operatorname{vol}(B_x))^2 \,\mathrm{d}x = \frac{1}{n^2} \int_0^1 \varphi_d(v)\, \delta(nv)^2 \,\mathrm{d}v, \tag{2}$$

where $\varphi_d(v)$ is the density of boxes with volume v: $\varphi_d(v) = \mathrm{d}\Phi_d(v)/\mathrm{d}v$, where $\Phi_d(v)$ is the volume of the set $\{x \in [0,1]^d\colon x_1 x_2 \cdots x_d \leqslant v\}$. The function $\Phi_d(v)$ can be calculated by induction on the dimension d. The calculation is not too difficult and we omit it. The result is that for $0 < v < 1$,

$$\varphi_d(v) = \frac{(-\ln v)^{d-1}}{(d-1)!}.$$

(I believe such a simple formula must be known and have a simple reason, but so far I could only get it via the more complicated function $\Phi_d(v)$.)

Equation (2) can be called a "nonintegrality lower bound." The integral does not seem to be expressible by a simple formula, but with some care its value can be approximated numerically even for large values of n.

A simpler approach is to pass to a "small-boxes lower bound," i.e. integrating up to $v_0 = 1/2n$ only:

$$D_2^*(n)^2 \geqslant \frac{1}{n^2} \int_0^{v_0} \varphi_d(v)\, \delta(nv)^2 \, \mathrm{d}v = \int_0^{v_0} \varphi_d(v)\, v^2 \, \mathrm{d}v = \frac{v_0^3}{3^d} \sum_{i=0}^{d-1} \frac{(-3 \ln v_0)^i}{i!}. \quad (3)$$

The last sum consists of the first d terms of the Taylor series for $e^{-3 \ln v_0} = v_0^{-3}$. Hence, for $(-\ln v_0)$ small, enough compared to d, i.e. for $\ln n$ not too large in terms of d, the right-hand side approaches 3^{-d}, showing that $D_2^*(P_0)$ is close to being the best possible.

Table I illustrates this phenomenon quantitatively. For selected dimensions, the largest values of n are shown for which $D_2^*(P_0)^2 = 3^{-d}$ exceeds the lower bounds (3) and (2) by at most a given factor (1.1, 2, and 10). The values for the small-boxes lower bound (3) are in the top rows, and the values for the nonintegrality lower bound (2) are in the bottom rows. (Since the function φ_d is given by an alternating series with possible near-cancellations, the calculations were performed in a 30-digit and 40-digit precision in the computer algebra system Mathematica 2.2.)

These observations indicate that for dimensions above 30, say, D_2^* is not a very good non-uniformity measure unless the considered point set is astronomically large. As an alternative, one may assign large weights to boxes of large volume. One such notion has been introduced recently by Hickernell [10] (with a different motivation, namely because of a better behavior for error bounds in numerical integration). This modified discrepancy is defined as the L_2-average of the L_2-discrepancies of all the $2^d - 1$ projections of P on the coordinate subspaces of dimensions 1, 2, ..., d. A simple calculation shows that thus modified L_2-discrepancy

TABLE I

Largest Values of n for Which the Squared L_2-Discrepancy of P_0 Differs from the Lower Bounds (3) and (2) by No More than a Given Factor.

	Dimension $d=$							
Factor	10	15	20	25	30	40	50	60
1.1	3	14	59	250	1000	$2.0 \cdot 10^4$	$4.1 \cdot 10^5$	$8.4 \cdot 10^6$
	3	17	71	300	1300	$2.5 \cdot 10^4$	$5.0 \cdot 10^5$	$1.0 \cdot 10^7$
2	12	66	350	1900	$1.0 \cdot 10^4$	$2.7 \cdot 10^5$	$7.7 \cdot 10^6$	$2.1 \cdot 10^8$
	15	82	440	2300	$1.2 \cdot 10^4$	$3.5 \cdot 10^5$	$9.7 \cdot 10^6$	$2.7 \cdot 10^8$
10	56	410	2800	$1.8 \cdot 10^4$	$1.2 \cdot 10^5$	$4.8 \cdot 10^6$	$1.9 \cdot 10^8$	$7.0 \cdot 10^9$
	76	540	3700	$2.4 \cdot 10^4$	$1.6 \cdot 10^5$	$6.3 \cdot 10^6$	$2.4 \cdot 10^8$	$9.1 \cdot 10^9$

Note. Values rounded to two significant digits.

of the pathological set P_0 is also nearly optimal for d large enough relative to n, but quantitatively, this effect is much weaker than for the usual L_2-discrepancy. Perhaps a yet more satisfactory notion could be obtained using Hickernell's generalization of L_2-discrepancy (see [10]) by a suitable choice of the optional parameters.

Remark. As was mentioned in the Introduction, asymptotically tight lower bounds for the L_2-discrepancy are known (formula (1)), first proved by Roth [24]. By inspecting Roth's proof (as presented in [1]), one can calculate that the lower bound is

$$D_2^*(P)^2 \geqslant \frac{c}{n^2}\, 2^{-4d} \binom{\lfloor \log_2 n \rfloor + d + 1}{d-1}, \tag{4}$$

where c is an absolute constant (something like 1/8; the precise bound depends on how carefully the proof is done).[2] Despite of being asymptotically tight for any fixed d, this bound seems to have little relevance for estimating the L_2-discrepancy for sets of a practically manageable size (up to 10^8, say). For instance, the asymptotically very weak small-boxes lower bound (3) is much better than (4) for up to 10^7 points in dimensions $\geqslant 6$. By some modifications of Roth's proof, the bound (4) can be improved somewhat, but so far I have not succeeded in getting considerably better values. Few other techniques exist for lower-bounding the L_2-discrepancy and its variants (Beck [1], Montgomery [14]), but so far they do not look very promising in this respect either.

3. SCRAMBLED NETS AND THEIR IMPLEMENTATION

A number of constructions of n-point sets $P \subset [0,1]^d$ are known with (worst-case) discrepancy $D^*(P) \leqslant (C_d/n)\log^{d-1} n$, where C_d is a constant dependent on the dimension d; let us refer to constructions with this property as *low-discrepancy sets*. This bound is believed to be asymptotically optimal as a function of n for every fixed d. Lot of work has been devoted to improving the theoretical estimate of C_d in terms of d (see e.g., [18] for an overview and [19] for a more recent result). Little is known about actual numerical values of discrepancy for these theoretical constructions (mainly because of the difficult computability of the worst-case discrepancy).

[2] An explicit bound calculated by Kuipers and Niederreiter [12, pp. 103–104] from their version of Roth's proof, which is $2^{-8d}(\log_2 n/(d-1))^{d-1}$, is considerably smaller and it seems to be unnecessarily pessimistic.

The knowledge about L_2-discrepancy is more satisfactory in several respects. As was remarked in the Introduction, the asymptotic order of magnitude is known precisely for d fixed, and several numerical studies for various point set constructions have been conducted. Several asymptotically optimal constructions have been published [25, 3, 6, 26]. Recently, Hickernell [10] derived an exact formula for the expected squared L_2-discrepancy for a certain class of randomized constructions (it would be interesting to make a similar theoretical investigation for other low-discrepancy constructions).

To recall the randomized constructions considered by Hickernell, we begin with a few definitions. Let $b \geqslant 2$ be a fixed integer. By a *b-ary canonical interval* we mean an interval of the form $[k/b^q, (k+1)/b^q)$, where $q \in \{0, 1, 2, ...\}$ and $k \in \{0, 1, ..., b^q - 1\}$. For instance, ternary canonical intervals are $[0, 1)$, $[0, 1/3)$, $[1/3, 2/3)$, $[2/3, 1)$, $[0, 1/27)$, $[1/27, 2/27)$,.... A *b-ary canonical box* is a Cartesian product of d b-ary canonical intervals. A set $P \subset [0, 1]^d$ is called a $(0, m, d)$*-net in base b*, where $m \geqslant 0$ is an integer, if $|P| = b^m$ and each b-ary canonical box of volume b^{-m} contains exactly one point of P. (In the literature, a more general notion of (t, m, d)-nets in base b is sometimes considered (see, e.g., [18]), but for our purposes the definition just given is sufficient.) We will say only "$(0, m, d)$-net" instead of "$(0, m, d)$-net in base b" if b is understood from context. If b and d are fixed then any $(0, m, d)$-net is a low-discrepancy set. A construction due to Faure [4] provides an $(0, m, d)$-net in base b for each d and m, with b being the first prime $\geqslant d-1$.

Randomization of $(0, m, d)$*-Nets.* For an approximate result computed by Monte-Carlo methods, such as a Monte-Carlo estimate of a multidimensional definite integral, the error can be estimated statistically by repeating the computation several times. In contrast, deterministic constructions of low-discrepancy sets, such as Faure's, provide only one set for a given size and dimension, and hence, only various theoretical worst-case error estimates are available. These are often unnecessarily pessimistic. Hence, it is advantageous to introduce some randomness into low-discrepancy constructions. Another heuristic reason for introducing randomness into the constructions are several theoretical asymptotic results. For instance, it can be shown that another well-known low-discrepancy set, the so-called Halton–Hammersley set [7, 8], does not attain an asymptotically optimal L_2-discrepancy. On the other hand, Roth [25] modified this construction by introducing an extra parameter (a certain cyclic shift) into it, and he proved that the expected L_2-discrepancy for a random choice of this parameter already achieves the asymptotically optimal bound. Also all other known constructions of sets with asymptotically optimal L_2-discrepancy involve, to my knowledge, some randomization.

Owen [21] discusses a very general method that allows one to "randomize" many of the known constructions of low-discrepancy sequences (while retaining the low-discrepancy property). He also shows that such a randomization yields better theoretical estimates for approximating integrals of sufficiently smooth functions and that the integration error can be estimated statistically by repeated experiments.

Owen's method can be described as follows. Let $b \geqslant 2$ be an integer. We are going to describe a mapping σ of the interval $[0, 1)$ onto itself; any σ arising by the procedure described below is called a *b-ary scrambling*. Let $x \in [0, 1)$ be a real number. Write $x =_b 0.a_1 a_2 a_3 \ldots$, meaning that $0.a_1 a_2 a_3 \cdots$ is the b-ary representation of x, that is, $x = a_1 b^{-1} + a_2 b^{-2} + a_3 b^{-3} + \cdots$, $a_j \in \{0, 1, \ldots, b-1\}$. To determine $\sigma(x)$, we first fix some permutation π of the set $\{0, 1, \ldots, b-1\}$, and we let the first b-ary digit of $\sigma(x)$ be $\pi(a_1)$. Next, for each possible value of a_1, we fix a permutation π_{a_1} of $\{0, 1, \ldots, b-1\}$, and we define the second b-ary digit of $\sigma(x)$ as $\pi_{a_1}(a_2)$. Continuing analogously, we fix b^2 permutations π_{a_1, a_2} for all possible choices of a_1 and a_2, and let the third b-ary digit of $\sigma(x)$ be $\pi_{a_1, a_2}(a_3)$, etc; in general, we have $\sigma(x) =_b 0.b_1 b_2 b_3 \cdots$ with $b_j = \pi_{a_1, a_2, \ldots, a_{j-1}}(a_j)$. This finishes the definition of a b-ary scrambling. A b-ary scrambling is in fact a bijective mapping, up to a countable set of exceptional numbers (problems arise with numbers having all digits equal to $b-1$ from some position on).

If all the (countably many) permutations π, π_0, π_1, , ..., π_{b-1}, $\pi_{0,0}$, $\pi_{0,1}$, ..., $\pi_{0,b-1}$, $\pi_{1,0}$,... defining σ are chosen independently at random, we say that σ is a *fully random b-ary scrambling*.

Let $P \subset [0, 1]^d$ be a $(0, m, d)$-net in base b, and let

$$\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_d)$$

be a d-tuple of b-ary scramblings. We define the set

$$\boldsymbol{\sigma}(P) = \{(\sigma_1(p_1), \sigma_2(p_2), \ldots, \sigma_d(p_d)) \colon p \in P\},$$

where p_k stands for the kth coordinate a point $p \in P$. It is easy to check that for any $\boldsymbol{\sigma}$, $\boldsymbol{\sigma}(P)$ is again a $(0, m, d)$-net. If $\sigma_1, \ldots, \sigma_d$ are chosen as fully random and mutually independent b-ary scramblings, we say that $\boldsymbol{\sigma}(P)$ arises from P by a *fully random scrambling*.

Hickernell [10] derived an exact formula for the expected squared L_2-discrepancy of $\boldsymbol{\sigma}(P)$, where $\boldsymbol{\sigma}(P)$ arises from an arbitrary fixed $(0, m, d)$-net P by a fully random scrambling. In fact, his proof does not use the whole power of a fully random scrambling; for the validity of the formula, it is enough that the vector $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_d)$ of b-ary scramblings is sampled from a probability distribution satisfying several axioms listed in [10, Assumption 1.2]. The formula in [10] is given for a generalized notion of L_2-discrepancy; specialized to the particular case of the "usual" L_2-discrepancy, it reads[3]

$$\mathbf{E}D_2^*(\boldsymbol{\sigma}(P))^2 = \frac{1}{2^d} \sum_{\ell=1}^{d} \binom{d}{\ell} \left(-\frac{1}{3}\right)^{\ell} \left[-1 + \left(1+\frac{1}{b}\right)^{\ell} \left(1-\frac{1}{b}\right) \right.$$
$$\left. \times \sum_{t=0}^{m-1} \binom{\ell+t-1}{\ell-1} b^{-2t} \sum_{r=0}^{m-t-1} \binom{\ell-1}{r} \left(-\frac{1}{b}\right)^{r} \right], \quad (5)$$

where $\mathbf{E}$ denotes the expectation with respect to a random choice of $\boldsymbol{\sigma}$ and P is an arbitrary fixed $(0, m, d)$-net in base b.

As numerical comparisons in Section 4 below witness, fully randomly scrambled b-ary nets are quite good in comparison with other low-discrepancy set constructions. Also Owen's results [21] indicate the usefulness of fully random scrambling. However, a computer implementation of a fully random scrambling is somewhat problematic (it will be briefly discussed at the end of this section). Here we show that the formula (5) remains valid under assumptions weaker than those in [10] on the distribution $\boldsymbol{\sigma}$ is sampled from. We point out several types of random scrambling which satisfy these assumptions and which can be implemented very efficiently.

First we formulate the modified axioms. In the interest of simplicity, we are not aiming at the greatest possible generality, so that the modified axioms are also stronger in some respect than those in [10]. Let P be a fixed $(0, m, d)$-net in base b, and let $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, ..., \sigma_d)$ be a random variable whose values are vectors consisting of b-ary scramblings:

A. Each of the b-ary scramblings $\sigma_1, \sigma_2, ..., \sigma_d$ is sampled from the same distribution $\mathcal{D}$, and these samplings are mutually independent. (This implies Hickernell's Assumption 1.2b.)

B. If $x \in [0, 1)$ is any real number and σ is randomly drawn from the distribution $\mathcal{D}$, then $\sigma(x)$ is uniformly distributed in $[0, 1)$. (This is a counterpart of Hickernell's Assumption 1.2a; I believe that his formulation of 1.2a is not quite appropriate for his proof.)

B′. (A weakening of B). If $x \in [0, 1)$ is any real number and σ is randomly drawn from the distribution $\mathcal{D}$, then the first two moments of $\sigma(x)$

[3] Warning. This formula should be evaluated with a high-precision arithmetic, because the "-1" term nearly cancels with the other term in the square brackets.

coincide with those of the uniform distribution on $[0, 1]$, that is, $\mathbf{E}\sigma(x) = 1/2$ and $\mathbf{E}\sigma(x)^2 = 1/3$.

C. Let

$$x =_b 0.a_1 a_2 \cdots a_r a_{r+1} a_{r+2} \cdots \qquad \text{and} \qquad x' =_b 0.a_1 a_2 \cdots a_r a'_{r+1} a'_{r+2} \cdots$$

be two real numbers with $a_{r+1} \neq a'_{r+1}$; i.e., x and x' share r first b-ary digits but no more. Write $\sigma(x) =_b 0.b_1 b_2 b_3 \cdots$, $\sigma(x') =_b 0.b'_1 b'_2 b'_3 \cdots$, where σ is randomly drawn from the distribution $\mathcal{D}$. Then

(i) $b_1 = b'_1$, $b_2 = b'_2$, ..., $b_r = b'_r$ (this follows from the definition of a b-ary scrambling),

(ii) the pair (b_{r+1}, b'_{r+1}) is uniformly distributed on the set

$$\{(q, q') : q, q' \in \{0, 1, ..., b-1\}, q \neq q'\}, \tag{6}$$

and

(iii) for each $j > r+1$ and for each pair (q, q') of possible values of (b_{r+1}, b'_{r+1}), the conditional expectation $\mathbf{E}[b_j - b'_j \mid b_{r+1} = q, b'_{r+1} = q'] = 0$.

Proposition 3.1. *Let P be a fixed* $(0, d, m)$*-net in base b, and suppose that $\boldsymbol{\sigma} = (\sigma_1, ..., \sigma_d)$ is a random vector of b-ary scramblings satisfying conditions A–C above. Then the formula in* [10] *for the expected value of the squared generalized L_2-discrepancy of $\boldsymbol{\sigma}(P)$ remains valid. In particular,* (5) *holds* (*for the usual L_2-discrepancy*), *and it does so even if the condition B is replaced by B'.*

Proof. Since most of the proof is identical to Hickernell's proof and only a small modification is required, we assume familiarity with Section 2 of [10]. One can check that conditions A and B are sufficient for the whole proof in [10] except for Lemma 2.4 (and also that if the optional function μ in [10] is given by a quadratic polynomial then B′ suffices instead of B). In Lemma 2.4 of [10], only the following claim has to be re-proved.

Claim. Let x, x' be real numbers as in condition C above, i.e. sharing exactly r first b-ary digits but no more. Then,

$$\mathbf{E}\, |\sigma(x) - \sigma(x')| = \frac{1}{3}\left(1 + \frac{1}{b}\right) b^{-r},$$

where the expectation is over σ randomly drawn from the distribution $\mathcal{D}$.

Proof of the claim. Writing $\sigma(x) =_b 0.b_1 b_2 b_3 \cdots$, $\sigma(x') =_b 0.b'_1 b'_2 b'_3 \cdots$, we have

$$|\sigma(x) - \sigma(x')| = b^{-(r+1)} \, |b_{r+1} - b'_{r+1}| + \xi(b'_{r+1} - b_{r+1}) \cdot \sum_{j=r+2}^{\infty} b^{-j} (b_j - b'_j),$$

where $\xi(t) = 1$ for $t \geqslant 0$ and $\xi(t) = -1$ for $t < 0$. From (ii) in condition C we get that the expectation $\mathbf{E}\, |b_{r+1} - b'_{r+1}|$ equals $(b+1)/3$ (as calculated in [10]). Finally, by condition C(iii), the expectation of $\xi(b'_{r+1} - b'_{r+1})(b_j - b'_j)$ is 0 conditioned on $(b_{r+1}, b'_{r+1}) = (q, q')$ for any (q, q'). Hence, $\mathbf{E}[\xi(b'_{r+1} - b'_{r+1})(b_j - b'_j)] = 0$ as well, and the claim follows. ▮

Let us remark that condition C could be weakened in various ways (all we need is to derive the claim), but the present formulation seems convenient enough, at least for all the examples below.

Examples of efficiency implementable random scramblings.

Here we list several distributions $\mathscr{D}$ of b-ary scramblings satisfying conditions B and C. Condition A can then be used as a definition, giving rise to a method of random scrambling of $(0, m, d)$-nets.

- *Random digit-scrambling.* For $x =_b 0.a_1 a_2 a_3 \cdots$, put $\sigma(x) =_b 0.b_1 b_2 b_3 \cdots$ with $b_j = \pi_j(a_j)$, where $\pi_1, \pi_2, \ldots$ are mutually independent random permutations of $\{0, 1, \ldots, b-1\}$. Such an σ will be called a *random b-ary digit-scrambling*. Conditions B and C are straightforward to verify; for C(iii), note that π_j is independent of π_{r+1}, and if $a_j = a'_j$ then $\pi_j(a_j) - \pi_j(a'_j) = 0$ always while for $a_j \neq a'_j$, $\pi_j(a_j) - \pi_j(a'_j) = t$ has the same probability as $\pi_j(a_j) - \pi_j(a'_j) = -t$, for any number t.

Let us remark that the idea of applying independent permutations on the b-ary digits in each coordinate is contained e.g., in Niederreiter's definition of a general class of (t, m, d)-nets in base b [18].

How do we implement a random digit-scrambling? For all practical purposes, it is sufficient to round the coordinates of points to some $m_0 \geqslant m$ digits in base b, with m_0 being a small number. Then it is enough to generate and store m_0 random independent permutations of $\{0, 1, \ldots, b-1\}$ per one random digit-scrambling, which requires $O(bm_0)$ memory. Moreover, for most constructions (such as Faure's), only the first m b-ary digits of the coordinates are nonzero, and so it suffices to store only m permutations plus one sequence of $m_0 - m$ random b-ary digits.

- *Random linear digit-scrambling.* The permutations $\pi_1, \pi_2, \ldots$ above, defining a digit-scrambling σ, need not be chosen from the set of all permutations of $\{0, 1, ..., b-1\}$. Assuming that b is a prime, one can restrict oneself to permutations of the form $\pi_j(a) = h_j a + g_j$, where $h_j \in \{1, 2, ..., b-1\}$ and $g_j \in \{0, 1, ..., b-1\}$ and the arithmetic is modulo b. If the h_j's and g_j's are chosen uniformly and independently at random, we speak of a *random linear digit-scrambling*.

Verifying conditions B and C is not difficult, and it follows considerations made in standard constructions of pairwise independent random variables. (These are used, for instance, for limiting the amount of randomness in algorithms; see e.g., [13] or [17] for an introduction and references.) Concerning the condition B, we note that for any fixed x and any fixed choice of h_j, the b possible values for g_j are in a bijective correspondence with the b possible values of the digit b_j of $\sigma(x)$. From this condition B follows. As for C(ii), we observe that for arbitrary $a_{r+1} \neq a'_{r+1}$ and for any choice of the pair (q, q'), $q \neq q'$, there exists exactly one pair (h_{r+1}, g_{r+1}) with $h_{r+1} \neq 0$ satisfying the linear system $h_{r+1} a_{r+1} + g_{r+1} = q$, $h_{r+1} a'_{r+1} + g_{r+1} = q'$ (in the b-element field). Finally for C(iii), since (h_j, g_j) is independent of (h_{r+1}, g_{r+1}), the conditioning on $(b_{r+1}, b'_{r+1}) = (q, q')$ can be omitted. We again distinguish the cases $a_j = a'_j$ and $a_j \neq a'_j$. In the former case $b_j = b'_j$ always, and in the latter case, the the pair (b_j, b'_j) is uniformly distributed on the set (6), and hence $\mathbf{E}[b_j - b'_j] = 0$ follows too.

To represent a linear digit-scrambling, it is sufficient to store $2m_0$ numbers. All of this can also be generalized to the case when b is a prime power, but then one has to implement the arithmetic in the finite field $GF(b)$.

• *A further reduction of randomness.* More for theoretical than practical interest, we remark that if only condition B′ is required instead of B, the amount of randomness in a random linear digit-scrambling can be reduced further. We require that each g_j be uniformly distributed in $\{0, 1, ..., b-1\}$ and that $g_1, g_2, g_3, ...$ be *pairwise independent*, meaning that for each $j \neq j'$, g_j and g'_j are independent random variables. Moreover, we let $h_1 = h_2 = \cdots = h$, where h is a random variable uniformly distributed in $\{1, 2, ..., b-1\}$ and mutually independent with $(g_1, g_2, ...)$.

First we verify B′. For each fixed value of h, the b-ary digits of $\sigma(x) =_b 0.b_1 b_2 b_3 ...$ are uniformly distributed and pairwise independent as well, and so

$$\mathbf{E}\sigma(x)^2 = \mathbf{E}\left(\sum_{j=1}^{\infty} b^{-j} b_j\right)^2 = \sum_{j=1}^{\infty} b^{-2j}\, \mathbf{E} b_j^2 + 2 \sum_{1 \leqslant j < j'} b^{-(j+j')}\, \mathbf{E} b_j b_{j'}.$$

The last expression is the same for b_j's fully independent as for b_j's pairwise independent, and so $\mathbf{E}\sigma(x)^2 = 1/3$.

Condition C(ii) is checked exactly as before. In C(iii), we observe that if h and g_{r+1} are fixed, then b_{r+1} and b'_{r+1} are fixed as well, while g_j is still random. Then $(b_j, b'_j) = (ha_j + g_j, ha'_j + g_j)$ is uniformly distributed over all pairs $(u, u+u_0)$, where the addition is modulo b, $u_0 = h(a'_j - a_j)$ is fixed, and $u \in \{0, 1, ..., b-1\}$. Hence, $\mathbf{E}[b_j - b'_j \mid h, g_{r+1}] = 0$ and C(iii) holds.

It is possible to sample the values of m pairwise independent random variables uniformly distributed on $\{0, 1, ..., b-1\}$ by generating $O(\log m + \log b)$ random bits (see, e.g., [13] for more information and references on constructions of pairwise independent random variables). Hence, if we work with m_0 b-ary digits, a random scrambling still sufficient to attain the expected squared L_2-discrepancy given by (5) can be represented by only $O(\log m_0 + \log b)$ bits of information (per coordinate of the set being scrambled).

• *Random linear scrambling*. Limiting the amount of randomness in a scrambling too much might make the properties of the scrambled set worse in practice, even if the expected L_2-discrepancy is preserved (since L_2-discrepancy is only one of the measures of quality of a point set). Here is another way of random scrambling, inspired by Tezuka [29], which might perhaps perform better in practice than the random linear digit-scrambling, as it scrambles the set "more thoroughly." If $x =_b 0.a_1a_2a_3 \cdots \in [0, 1)$ is a real number then we set $\sigma(x) =_b 0.b_1b_2b_3 \cdots$ with

$$b_j = \sum_{i=1}^{j} h_{ij} a_i + g_j. \tag{7}$$

The arithmetic in this formula is modulo b, the g_j's and the h_{ij}'s with $i < j$ are chosen randomly and independently from $\{0, 1, ..., b-1\}$, and the h_{jj}'s are chosen randomly and independently from $\{1, 2, ..., b-1\}$. Verifying the conditions B and C is very similar to the case of random linear digit-scrambling and we omit it. Let us remark that Tezuka's method ("generalized Faure sequences" in [29]) is essentially equivalent to setting

$$b_j = \sum_{i=1}^{j} h_{ij} a_i$$

with the h_{ij}'s chosen as before (but with the additive terms g_j omitted). For this method, there are difficulties with analyzing the expected L_2-discrepancy because of the special role played by zero digits. Introducing the additive terms g_j makes the situation much simpler and more regular.

The implementation of Faure's construction with random linear scrambling remains essentially the same as the implementation of the generalized Faure's construction described in [29]. The above described constructions of random scramblings are fast and simple to implement. For constructions of $(0, m, d)$-nets in base b, the sequence of b-ary digits for each coordinate is produced as an intermediate result anyway, so the scrambling amounts to a few simple operations with the digits before they are converted to the numeric value. The practical suitability of the various suggested ways of random scrambling remains to be judged (by computational tests in various applications and/or by theoretical analysis of other characteristics of the scrambled sets than the L_2-discrepancy).

On the Implementation of a Fully Random b-ary Scrambling

The implementation of a fully random b-ary scrambling seems somewhat problematic, since a large amount of information must be stored to represent the scrambling. Still, we outline a reasonably usable implementation. First, we observe that since no two points of a $(0, m, d)$-net P share more than m first b-ary digits, a fully random scrambling produces coordinates whose b-ary representation is $0.b_1 b_2 \cdots b_m v_1 v_2 v_3 \cdots$, where the sequence of b-ary digits $v_1, v_2, v_3, \ldots$ is chosen randomly and independently for each point of P and each coordinate separately (and thus it need not be stored). Hence, it suffices to store the permutations π, $\pi_{a_1}, \ldots, \pi_{a_1, \ldots, a_{m-1}}$ for all possible values of $a_1, a_2, \ldots, a_{m-1}$. It will be convenient to imagine that these permutations are indexed by the nodes of a complete b-ary tree of depth m (π sits at the root, $\pi_0, \pi_1, \ldots, \pi_{b-1}$ are at its b sons, etc.). Unlike random digit-scrambling, the number of these permutations is comparable to the number of points of the set being scrambled, and this may be prohibitively large (in most applications, the points are generated one by one and never stored all at once).

If the available storage is large enough, we can afford to generate all the required permutations, but otherwise we have to save memory. We describe a trick for doing this that trades speed of the generation for memory. The trick is based on the fact that the random numbers used for generation of the random permutations in practice come from some pseudorandom number generator. Pseudorandom generators produce a sequence of numbers by a fully deterministic computation (see, e.g., [29] for a discussion of modern pseudorandom generators). The continuation of such a sequence of pseudorandom numbers from any given moment on is uniquely determined by the current status of the generator. The status is usually stored in a few bytes of computer memory (say in 10 to several hundred bytes). If the current status is recorded and, later on, the generator is reset to the same status, it will repeat the same sequence of numbers.

In our situation, we can first generate and store the permutations corresponding to the first m_1 levels of the b-ary tree. Here $m_1 \leqslant m$ is a parameter that can be set at will, so as to achieve a suitable trade-off between speed and storage. Then we do "as if" we generated the permutations in the subtrees attached to nodes at level m_1, handling them one by one. For each node at level m_1, we record the status of the pseudorandom generator at the moment when the generation of permutations in the corresponding subtree started. The permutations in these subtrees are not stored. Next, if we want to compute $\sigma(x)$, where $x = 0.a_1 a_2 \cdots a_m$ in b-ary and σ is our random b-ary scrambling, we first compute the first m_1 digits using the stored permutations. Then we restore the pseudorandom generator to the appropriate state, and we re-generate the permutations from the corresponding subtree for finding the remaining digits of $\sigma(x)$.

This method is not too complicated and works reasonably well (with some extra lower-level tricks; e.g., we extract several random b-ary digits from one pseudorandom real number, etc.). It is an "honest" implementation of a fully random b-ary scrambling, at least as random as one can get with currently available pseudorandom generators. On the other hand, from a practical point of view, one can hope that the much simpler scrambling methods suggested above might work satisfactorily enough in applications (this remains to be tested). Another possibility is a combination of the fully random scrambling for the first m_1 digits and a random digit-scrambling for the remaining digits, say.

4. NUMERICAL COMPUTATIONS

Sets versus Sequences

Here we present some numerical studies of L_2-discrepancy for several low-discrepancy constructions. We should begin with a terminological remark. For some constructions of low-discrepancy sets (as discussed in Section 3), the number of points has to be of a special form (for example, b^m for Faure's construction). In practical applications, it seems preferable to be able to choose the set size freely. Also, sometimes one would like to add a few more points to the set already used and see how the result changes, say. For such reasons, the constructions used in applications are usually formulated as infinite sequences. That is, a rule is given for producing an infinite sequence $(x_1, x_2, x_3, ...)$ of points in $[0, 1]^d$, such that the discrepancy of each initial segment $\{x_1, x_2, ..., x_n\}$ is $O(\log^d n/n)$ (let us call such a sequence a *low-discrepancy sequence*).

Let us mention that any construction of a low-discrepancy sequence in dimension d yields a low-discrepancy set in dimension $d+1$ for each n, and conversely, a construction of low-discrepancy sets (for all n) in dimension $d+1$ provides a construction of a low-discrepancy sequence in dimension d. For the reader's convenience, we recall the idea of a proof. To obtain an n-point set in $\mathbb{R}^{d+1}$ from a sequence $(x_1, x_2, ...)$ in $\mathbb{R}^d$, we append the $(d+1)$th coordinate i/n to x_i, $i = 1, 2, ..., n$. For the reverse direction, let $P_i \subset [0, 1]^{d+1}$ be a low-discrepancy set of size $n_i = 2^{2^i}$. We order the points of P_i by the $(d+1)$th coordinate and then we delete the last coordinate of each point, obtaining an n_i-term sequence in $\mathbb{R}^d$. Finally, by concatenating all these finite sequences for $i = 1, 2, ...$, a low-discrepancy infinite sequence results. Hence the difference between sets and sequences is merely a technical one and both concepts are essentially equivalent (with the dimension jump of 1). In the rest of this section, we will consider constructions of infinite sequences (as opposed to Section 3).

The Tested Constructions

Two basic constructions we consider here are the *Halton sequence* [7] and the *Faure sequence* [4]. Halton's construction is very simple and we recall it briefly. Let $2 = p_1 < p_2 < \cdots < p_d$ be the d smallest primes. For an integer i and index $k \in \{1, 2, ..., d\}$, let $i =_{p_k} a_m a_{m-1} \cdots a_1$ be a representation of i in base p_k, i.e. $i = \sum_{j=1}^{m} a_j p_k^{j-1}$. Then we define the kth coordinate of the ith point x_i of the Halton sequence by $(x_i)_k =_{p_k} 0.a_1 a_2 \cdots a_m$. The construction of the Faure sequence is somewhat more complicated and we refer to the literature (e.g., [1]).

Both the Halton and Faure sequences are known to be low-discrepancy ones in the above-defined sense (proofs can be found in many sources, e.g., [1]). The theoretical bound for the discrepancy in terms of d is considerably better for the Faure sequence than for the Halton sequence (see [18]). On the other hand, numerical experiments indicate that for up to 2^{16} points, Halton's construction tends to have a slightly (and sometimes significantly) better L_2-discrepancy than Faure's one.

The idea of random b-ary scrambling can be applied to the Halton and Faure sequences as well. Namely, the Faure sequence remains a low-discrepancy one if we apply a b-ary scrambling σ_k on the kth coordinates of all points (the same scrambling for all points), $k = 1, 2, ..., d$, and, similarly, applying a p_k-ary scrambling to the kth coordinates of points of the Halton sequence preserves the low-discrepancy property. Using the names of the various random scramblings introduced in section 0, we can speak of *fully randomly scrambled Faure sequence*, *randomly digit-scrambled Halton sequence*, etc. Moreover, we also consider the *generalized Faure sequences* as in [29].

It does not seem obvious how to extend Hickernell's considerations in [10] to find a formula for the expected squared L_2-discrepancy for initial segments of randomly scrambled sequences (or at least such a computation looks quite complicated). On the other hand, it is easy to see from Hickernell's proof and the considerations in section 0 that for any given set P, all the scrambling methods satisfying conditions A–C give the same expected squared L_2-discrepancy. The expectation for the generalized Faure method may differ slightly.

All the low-discrepancy sequence constructions discussed so far belong to a common family of "constructions based on b-ary expansion" ("digital nets" in the terminology of [18]). Another, quite different type of constructions are "lattice-based" ones. For comparison, we have selected just one construction of this family, due to Richtmyer [23]: The kth coordinate of the ith point of this sequence is $(i\sqrt{p_k}) \bmod 1$, where p_k is the kth prime as in Halton's construction. For an arbitrary dimension $d > 2$, this sequence is not known to be a low-discrepancy one, but it seems to be quite popular and it is very simple to implement. James *et al.* [11] report that it behaved better in dimensions about 15 than the other constructions they tested. There are many more lattice-based construction, such as the method of *good lattice points* (see [18]) or the method of Frolov [6] generalized by Skriganov [26], but these seem to be much less popular so far, perhaps because they appear much more complicated for implementation.

Excluding a Few Initial Points

As was observed by several researchers, studies of the L_2-discrepancy for the Halton sequence and similar ones can be significantly distorted by the first few points of the sequence (for instance, Morokoff and Caflisch [15] put a particular emphasis on this effect). For instance, since the first point of the Halton sequence lies quite near the origin, it causes many boxes of small volume to have a discrepancy of about $1/n$. Although this effect is insignificant asymptotically for d fixed and $n \to \infty$, it can become dominating for the L_2-discrepancy of the whole set if n is not very large in terms of d (this is a phenomenon similar to the lower bound based on empty boxes discussed in Section 2). To avoid this, all the calculations below were done for sequences with the first 100 points omitted (the choice of the parameter 100 being rather arbitrary), i.e. an n-point set was obtained as $\{x_{101}, x_{102}, ..., x_{n+100}\}$, where $(x_1, x_2, x_3, ...)$ is the underlying infinite sequence.[4] Furthermore, the L_2-discrepancy is computed with respect to

[4] This might be a reason for a significant disagreement of the numerical results presented below with those given by Tezuka [29]. For instance, while he finds the L_2-discrepancy of the Halton sequence in higher dimensions significantly worse than for various other constructions and, sometimes, even much worse than for random sets, we get that the same sequence with first 100 points omitted behaves quite well, certainly never significantly worse than random sets.

boxes anchored at $(1, 1, ..., 1)$, i.e. as if each point x were replaced by $(1, 1, ..., 1) - x$. For the randomly scrambled sequences, reasons for such precautions more or less disappear, but for the sake of uniformity we proceeded in the same way for all tested sequences.

Calculations Performed

The numerical tests performed involved the Halton and Faure sequences, their fully randomly scrambled and randomly digit-scrambled versions, the generalized Faure sequences according to [29], the Richtmyer sequence, and uniformly distributed random points. (The Halton, Faure, Richtmyer, and random sequences have been tested before, and here we include them for comparison.) The L_2-discrepancy for dimensions 10 and higher was computed using Warnock's algorithm [31] and for smaller dimensions by Heinrich's algorithm [9]; all discrepancy calculations were done in double precision arithmetic.

The code was written in the C++ language and run on several platforms using two different compilers. Random numbers were generated using the code for a pseudorandom number generator `CombLS2` published in Tezuka's book [29]. For comparison, some runs were repeated using the Unix `drand48` generator and using the other generator `CombMRG` in [29]; no significant differences appeared.

For studying the behavior of the L_2-discrepancy, we have three parameters to consider: the construction used, the dimension, and the number of points. Moreover, constructions involving randomness require at least several repetitions to make the results meaningful. It seems quite nontrivial to select a reasonable number of combinations of these parameters for calculations and to present them in a suggestive manner (this task has been solved in different ways by different authors—see [29, 15, 11] for a sample of possible approaches).

Preliminary calculations indicated that the dependence of the L_2-discrepancy on the dimension and on the number of points is reasonably smooth for the tested randomized constructions. The deterministic constructions (especially Faure's construction) show more significant oscillations, but qualitatively the behavior remains similar in various dimensions. For more detailed calculations, dimensions $d = 4$, 6, 10, 15, and 20 were chosen as a (hopefully representative enough) sample, and the numbers of points were set to $n = 2^{10}$, 2^{12}, 2^{14}, and 2^{16}. Control calculations done for several other n's chosen at random in the vicinity of the above-mentioned powers of 2 give a very similar overall picture.

The maximum value of n considered, $2^{16} \approx 6.5 \cdot 10^4$, results from practical limitations of both computation time and used memory. For instance, one discrepancy calculation for 2^{16} points in dimension 20 took several hours on a 100-MHz Pentium machine, and storing a point set of this size (in single precision) requires over 5 MB memory. Doing similar computations for significantly more points (about 10^6, say, which is a quite conceivable number for numerical integration) in a reasonable time would probably require a fast massively parallel machine (or a novel fast algorithm!).

Figure 1 summarizes the results of the calculations. It is divided into fields corresponding to the considered combinations of d and n. Each vertical column in each field corresponds to one tested construction (the constructions are distinguished by various line styles). The constructions are labeled by more or less self-explanatory labels (MonteCarlo stands for random uniformly distributed points, SHalton for the fully randomly scrambled Halton sequence, DSHalton for the randomly digit-scrambled Halton sequence, GFaure for the generalized Faure sequence, etc.). The vertical axis (drawn in a logarithmic scale) corresponds to the squared L_2-discrepancy; the value is normalized by dividing it by the expected squared L_2-discrepancy for a random point set. The value 1, corresponding thus to the expectation for a random set, is marked by a horizontal dotted line. The small horizontal bars drawn on the right-hand side of the construction's columns record results of individual L_2-discrepancy calculations. For constructions involving randomness, values for 30 repetitions are plotted. Although they mostly cannot be distinguished individually in the picture, they give some idea about the distribution. The horizontal bar on the left is the mean of these 30 results. A full vertical segment on the left shows the standard deviation (more precisely, the segment goes up to $\mu+\sigma$ and down to $\mu-\sigma$, where μ is the mean and σ is the standard deviation). For a mutual comparison of the results for various dimensions, note that the scale on the vertical axis is different for dimensions 4 and 6 than for the others.

Discussion

The picture clearly illustrates two main tendencies: The improvement in the L_2-discrepancy compared to a random set becomes more significant with increasing n for all tested constructions, but this effect diminishes with increasing dimension. For the largest tested n, no construction gave values better than some 70% in dimension 15, and in dimension 20, the improvement is still much more modest.

The basic, nonrandom constructions (Halton and, in particular, Faure) may behave in a considerably irregular manner. The Faure construction sometimes gives quite bad values in higher dimensions (e.g., for $d=15$, $n=2^{16}$), although it is quite good at other times. The Halton sequence seems to deviate from its randomized counterparts to a very good L_2-discrepancy more often than to worse one.

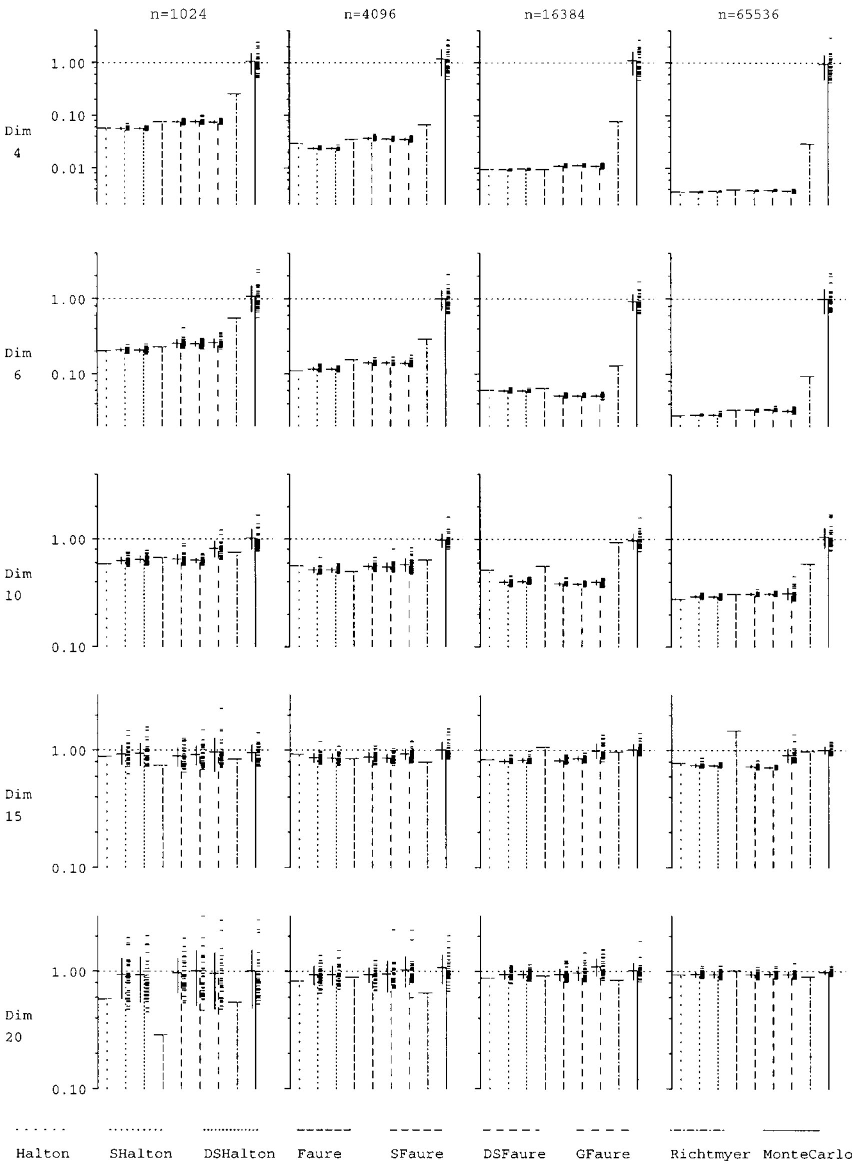

FIG. 1. Results of L_2-discrepancy computations.

The Richtmyer sequence is typically much worse than the other constructions in small dimensions (up to 10), although it never went much over the expectation for a random set (as the, e.g., the Faure sequence often does). In dimension 20, it does remarkably well.

All the tested randomization methods seem suitable for averaging out the fluctuations of the basic constructions. The differences between the Halton-based and Faure-based methods are quite small (a general tendency seems to be that the various scrambled Halton sequences are better than the Faure ones, but exceptions exist as well, e.g., $d=6$, $n=2^{14}$). The variance for all methods is small for dimensions up to 10, and it decreases with an increasing number of points. Also variances show no significant differences among the methods; one can say that, among the scrambled Faure-based methods, the generalized Faure sequences usually show the largest variance.

5. RUNNING TIME OF HEINRICH'S ALGORITHM

The Algorithm

For the reader's convenience, we briefly recall Heinrich's algorithm first. It will be formally more convenient to work with point sequences rather than with sets. Let $P=(x_1, x_2, ..., x_n)$ be a sequence of n points in $[0, 1]^d$, and let x_{ij} denote the jth coordinate of x_i. The L_2-discrepancy of P can be expressed as [31]

$$D_2^*(P)^2=\frac{1}{3^d}-\frac{2}{2^d n}\sum_{i=1}^{n}\prod_{k=1}^{d}(1-x_{ik}^2)+\frac{1}{n^2}\sum_{i,j=1}^{n}\prod_{k=1}^{d}(1-\max(x_{ik}, x_{jk})).$$

This immediately gives an algorithm for evaluating the L_2-discrepancy using $O(dn^2)$ arithmetic operations (in this algorithm, and also in Heinrich's one discussed below, the computation should be done with a sufficiently high precision, since the terms in the formula are typically of a considerably larger magnitude than the resulting discrepancy). The first two of the three addends in the formula can even be computed in $O(dn)$ time, so in designing a more efficient algorithm, it suffices to concentrate on evaluating $\sum_{i,j=1}^{n}\prod_{k=1}^{d}(1-\max(x_{ik}, x_{jk}))$. By replacing each point $x_i\in P$ by $(1, 1, ..., 1)-x_i$ (this is a tiny modification compared to [9]), we get the formally slightly simpler expression

$$M_d(P)=\sum_{i,j=1}^{n}\prod_{k=1}^{d}\min(x_{ik}, x_{jk}).$$

The first idea in the algorithm is to generalize the problem as follows: Let $A=(x_1, x_2, ..., x_n)$ and $B=(y_1, y_2, ..., y_m)$ be two given point sequences in $[0,1]^d$. Moreover, assume that each point x_i is assigned a real weight v_i, and each y_j is assigned a real weight w_j. We want to calculate the expression

$$M_d(A,B)=\sum_{i=1}^{n}\sum_{j=1}^{m} v_i w_j \prod_{k=1}^{d} \min(x_{ik}, y_{jk}). \tag{8}$$

First, suppose that we had $x_{id} \leqslant y_{jd}$ for all i, j. Then the $\min(x_{id}, y_{jd})=x_{id}$ for all i, j, and so the factor x_{id} can be incorporated into the weight v_i. For this particular situation we thus get

$$M_d(A,B)=\sum_{i=1}^{n}\sum_{j=1}^{m} (x_{id}v_i) w_j \prod_{k=1}^{d-1} \min(x_{ik}, y_{jk})=M_{d-1}(\tilde{A}, \bar{B}), \tag{9}$$

where the points of $\bar{B}$ are those of B with the dth coordinate deleted and with the same weights as they had in B, and the points of $\tilde{A}$ also arise from the points of A by deleting the last coordinate but the weight of the ith point of $\tilde{A}$ is set to $\tilde{v}_i=x_{id}v_i$, $i=1, 2, ..., n$.

In general, the dth coordinates are not so nicely separated, of course, but the above observation leads to a recursive algorithm of a divide-and-conquer type. Let μ denote a median of the dth coordinates of the points of A, i.e. a number such that $|\{i \in \{1, 2, ..., n\}: x_{id} \leqslant \mu\}| \geqslant n/2$ and $|\{i \in \{1, 2, ..., n\}: x_{id} \geqslant \mu\}| \geqslant n/2$. Partition the (weighted) sequence A into two sequences A_L, A_R in such a way that A_L has $\lfloor n/2 \rfloor$ points, A_R has $\lceil n/2 \rceil$ points, and the dth coordinates of points in A_L are $\leqslant \mu$ while the dth coordinates of points in A_R are $\geqslant \mu$. Partition also B into B_L and B_R, letting B_L consist of the points of B with dth coordinate $\leqslant \mu$ and B_R of the remaining ones. (Such a partitioning can be executed in $O(n+m)$ time since μ can be computed in $O(n)$ time by a linear-time median-finding algorithm.) Then we have the formula

$$M_d(A,B)=M_d(A_L,B_L)+M_d(A_R,B_R)+M_{d-1}(\tilde{A}_L,\bar{B}_R)+M_{d-1}(\bar{A}_R,\tilde{B}_L). \tag{10}$$

Here $\bar{A}_R$ arises from A_R by deleting the dth coordinate of its points, with weights unchanged and similarly for $\bar{B}_R$. Also the points of $\tilde{A}_L$ and $\tilde{B}_L$ are obtained from those of A_L and B_L by deleting the dth coordinate. Moreover, a point x_i lying in A_L has weight $x_{id}v_i$ in $\tilde{A}_L$, and a point y_j in B_L has weight $y_{jd}w_j$ in $\tilde{B}_L$.

The algorithm works recursively using the formula (11). There are two base cases solved directly. First, if $|A|=1$, we calculate $M_d(A, B)$ directly by definition, in $O(dm)$ time. And second, if $d=0$, we calculate

$$M_0(A, B)=\sum_{i=1}^{n} \sum_{j=1}^{m} v_i w_j=\left(\sum_{i=1}^{n} v_i\right)\left(\sum_{j=1}^{m} w_j\right) \tag{11}$$

in $O(n+m)$ time. This is the method we will analyze below. Alternatively, the recursion can be stopped for $d=1$ rather than for $d=0$, using a non-recursive $O(n+m)$ time one-dimensional algorithm noted by Frank and Heinrich [5].[5] Thus, the modified recursive algorithm can be analyzed by a straightforward modification of the analysis given below. The difference in efficiency of both versions is not drastic (the modified algorithm for a d-dimensional set performs certainly no better than the original algorithm for a set of the same size in dimension $d-1$), and so we omit this part.

Remarks on a Practical Implementation

• If the goal is to compute $M_d(P)$ for a given set P, the recursive algorithm can be called with $A=B=P$. As remarked in [9], however, the case $A=B$ can receive a special treatment. Specializing (11) to the $A=B$ case, we get the formula

$$M_d(A)=M_d(A_L)+M_d(A_R)+M_{d-1}(\tilde{A}_L, \bar{A}_R).$$

This reduces the running time roughly by a factor of 2, as can be verified by a simple extension of the analysis given below. Similarly, evaluating $M_d(A, B)$ directly according to the formula (10) requires $(d+2)\,nm$ arithmetic operations, but for $A=B$ this can be easily improved to $(d+2)(n+1)\,n/2$, i.e. also by a factor of about 2. Hence, in the analysis of the algorithm and its comparison with the direct method, we do not consider this modification.

• The coordinates of the points can be stored in a single global $n \times d$ array. The input to the recursive procedure is represented by the current dimension, two arrays of indices (the lists of points of A and B), and two arrays of real numbers storing the current weight functions v and w. These arrays should be allocated dynamically (their exact size is not known in advance) but their allocation follows a simple stack discipline.

[5] The $O(m+n)$ running time bound for the Frank–Heinrich algorithm assumes that the points of A and B are sorted. In the higher-dimensional recursive algorithm, this can be arranged by an $O((n+m)\log(m+n))$ preprocessing of the input.

• The median μ need not be computed exactly; it is enough that A_L and A_R have roughly equal sizes. In my experimental implementation, for $n > 21$, a random sample of 21 elements of A is taken, its median is computed exactly, and it is used for dividing A into A_L and A_R. This appears to work well enough.

Running Time Analysis

By induction on d, it is not too difficult to prove that the running time of the described algorithm for computing $M_d(A, B)$ is at most $C_d(m+n)\log^d(m+n)$, with C_d depending on d [9]. (This reduces to $C_d(m+n)\log^{d-1}(m+n)$ if the improved one-dimensional algorithm is used as was mentioned above.) Here we derive a more precise formula, capturing the dependence on the dimension as well. For definiteness, we will count the number of arithmetic operations (additions and multiplications); we will ignore comparisons, bookkeeping operations, etc. (those only add a constant-factor overhead).

Let $T(d, n, m)$ denote the maximum number of arithmetic operations executed by the algorithm called for an n-point set A and an m-point set B in dimension d.

First, we modify the algorithm a little, in order to get nicer formulas. We observe that except for the base cases ($n=1$ or $d=0$), the algorithm only adds together the results computed by recursive calls. Instead, we introduce a global variable for summing up the results, and in the base cases, we add the computed value to this global variable. Then the only arithmetic operations done in the recursive part of the algorithm are in the computation of the new weights of some points. The three additions in the formula (11) are accounted for in the base cases.

Using (12), we then get $T(0, n, m) = m+n$, and for $n=1$ we obtain $T(d, 1, m) = (d+1)\,m+1 \leqslant (d+1)(m+1)$. If m_1 denotes the size of B_L and m_2 the size of B_R in (11), we get that $m_1 + \lfloor n/2 \rfloor$ operations are needed to compute the weights for $\tilde{A}_L$ and $\tilde{B}_L$. Hence

$$T(d, n, m) \leqslant \max_{m_1+m_2=m} \left(\left\lfloor \frac{n}{2} \right\rfloor + m_1 + T\left(d, \left\lfloor \frac{n}{2} \right\rfloor, m_1\right) + T\left(d, \left\lceil \frac{n}{2} \right\rceil, m_2\right) \right.$$
$$\left. + T\left(d-1, \left\lfloor \frac{n}{2} \right\rfloor, m_2\right) + T\left(d-1, \left\lceil \frac{n}{2} \right\rceil, m_1\right) \right). \tag{12}$$

In the rest of the analysis, we assume that $n = 2^k$ is a power of 2. By an obvious monotonicity in n, we can also get reasonably good estimates for the intermediate values of n.

To solve the recurrence, we define an auxiliary function

$$f_0(d, k) = \max_{m \geqslant 0} \frac{T(d, 2^k, m)}{2^k + m} + 1.$$

In terms of f_0, the bounds for $T(d, n, m)$ lead to the initial conditions $f_0(0, k) \leqslant 2$, $f_0(d, 0) \leqslant d + 2$, and to the recursion

$$f_0(d, k) \leqslant f_0(d, k-1) + f_0(d-1, k-1)$$

for $k, d \geqslant 1$. The last formula is obtained from (13) using the estimate

$$\frac{\lfloor n/2 \rfloor + m_1}{n + m} \leqslant 1. \tag{13}$$

Let $f(d, k)$ be the function defined by the three conditions just given for f_0 with equality (i.e. an upper bound for $f_0(d, k)$). Introducing the generating function

$$F(x, y) = \sum_{d, k \geqslant 0} f(d, k)\, x^d y^k,$$

we thus get

$$F(x, y) = (y + xy)\, F(x, y) + \sum_{d=0}^{\infty} (d+2)\, x^d.$$

This gives an explicit solution

$$F(x, y) = \frac{1}{1 - y(1+x)} \left(\sum_{d=0}^{\infty} (d+2)\, x^d \right) = \sum_{k=0}^{\infty} y^k (1+x)^k \left(\sum_{d=0}^{\infty} (d+2)\, x^d \right).$$

From this, we express

$$f(d, k) = \sum_{i=0}^{d} (d + 2 - i) \binom{k}{i}.$$

For computing the L_2-discrepancy, we are really interested in the $m = n$ case. From the above calculation for $n = 2^k$ we obtain $T(d, n, n) \leqslant 2S(d, d+2, k)\, n$, where we write for later use

$$S(d, r, k) = \sum_{i=0}^{d} (r - i) \binom{k}{i}.$$

So far we have an estimate for the worst possible behavior of the algorithm. (It does not make much sense to investigate the best possible behavior of the algorithm) In case B_L is always empty and $B_R = B$ in the recursion, say, the algorithm finishes much faster (especially if a test for $m = 0$ is included), but such a situation is extremely unlikely for uniformly distributed sets A, B.

Next, let us make a heuristic consideration indicating how the algorithm might behave for uniformly distributed sets A and B. In this case, it is reasonable to expect that both A and B are approximately halved in each recursion step, so that $n \approx m$ and $m_1 \approx n/2$. (In fact, experiments indicate that a slightly faster algorithm results if always the larger of the two sets A, B is split exactly in half, or the condition $|A_L| + |B_L| = \lfloor (m+n)/2 \rfloor$ is imposed, instead of halving A.) Let $T_h(d, n)$ denote the number of arithmetic operations of the algorithm for $n = m$ and under the (ideal) assumption that $m_1 = \lfloor n/2 \rfloor$ always holds throughout the recursion. Then we get $T_h(0, n) = 2n$, $T_h(d, 1) = d + 2$ for $d \geqslant 1$, and

$$T_h(d, 2^k) = 2^k + 2T_h(d, 2^{k-1}) + 2T_h(d-1, 2^{k-1}).$$

This time, a good substitution is $g(d, k) = T_h(d, 2^k)/2^k + 1$. By a method similar to that above, we obtain $T_h(d, n) = S(d, d+3, k)\, n - n$ for $n = 2^k$.

Discussion. Let us compare the calculated estimates for $T(d, n, n)$ and $T_h(d, n)$ with $(d+2)\, n^2$, i.e. the number of operations needed for a direct evaluation according to the formula (9). Recall that we only consider the case $n = 2^k$.

First we note that the difference between the worst-case bound and the heuristic formula for the average case is not very significant; clearly, $T(d, n, n)$ is never more than $2T_h(d, n)$. Also, experiments indicate that the quantity $T_h(d, n)$ agrees with the actual number of arithmetic operations quite well (for $A = B$ chosen uniformly at random). For n in range from 2^{10} to 2^{16} and for d from 2 up to 16, the actual number of arithmetic operations was always less than $T_h(d, n)$, and never less than 89% of $T_h(d, n)$. Typically it was about 5% smaller than $T_h(d, n)$.

For $k \leqslant d$, the expression $S(d, r, k)$ can be summed up to $2 \cdot 2^k (r - k/2)$. Hence in this case we have $T(d, n, n) \leqslant 2(d + 2 - k/2)\, n^2$, which for $d = k$ is about the same as $(d+2)\, n^2$ for the direct method and it becomes slightly worse for smaller k, but never more than twice. Similarly, for $k \leqslant d$, we have $T_h(d, n) = (d + 3 - k/2)\, n^2$. Here we do get a slight saving, but a direct calculation may be comparably fast in practice because of the other operations in the recursive algorithm (storage allocation, median selection, etc.).

For k much larger than d, the last term in the sum $S(d, d+2, k)$ becomes dominating, and hence, $T_h(d, n)$ is asymptotically equivalent to $(3k^d/d!)\, n$

d fixed and $k \to \infty$, while $T(d, n, n)$ behaves as $4k^d/d!$. But unless the dimension is very small, this limit asymptotic behavior cannot be observed for practically reasonable values of n.

For d large and $k \geqslant d$, the behavior of $T_h(d, n)$ can be qualitatively described as follows. For $k = (3/2)\, d$, say, the saving is still very moderate (by a small constant factor). This is because the distribution of the binomial coefficients $\binom{k}{i}$ is strongly concentrated around $i = k/2$, in an interval of width roughly $2\sqrt{k}$. The coefficients in this interval sum up to nearly 2^k, and they are multiplied by roughly $d/4$ in $S(d, d+3, k)$ for $k = \frac{3}{2}d$. The behavior of $T_h(d, n)$ changes rather abruptly around $k = 2d$. At this point, we still have only about $\sqrt{d}$ factor saving (since a constant fraction of the "large" binomial coefficients is multiplied by a factor of about $\sqrt{d}$). But for k only slightly larger, the saving factor starts growing exponentially with k and becomes really significant. Unfortunately, with the speed of contemporary computers, this region can only be reached for quite small dimensions. The following table shows the smallest k's for which $T_h(d, 2^k)$ is by factors 3,10, and 100 smaller than $(d+2)2^{2k}$, for several dimensions d:

Improvement	Diumension $d =$					
factor	3	5	10	15	20	25
3	8	11	18	25	32	38
10	11	15	24	33	42	51
100	16	20	32	43	53	64

ACKNOWLEDGMENT

I thank an anonymous referee for pointing out an error in a previous version of this paper and for numerous insightful comments.

REFERENCES

1. Beck, J., and Chen, W. L., "Irregularities of Distribution," Cambridge Univ. Press, Cambridge, 1987.
2. Beck, J., and Sós, V. T., Discrepancy theory, *in* "Handbook of Combinatorics," pp. 1405–1446, North-Holland, Amsterdam, 1995.
3. Chen, W. L., On irregularities of distribution II, *Quart. J. Math. Oxford Ser. (2)* **34** (1983), 257–279.
4. Faure, H., Discrepancy of sequences associated with a number system (in dimension *s*), *Acta Arith.* **41**, No. 4 (1982), 337–351. [French].
5. Frank, F., and Heinrich, S., Computing discrepancies of Smolyak quadrature rules, *J. Complexity* **12** (1996), 287–314, 1996.

6. Frolov, K. K., An upper estimate for the discrepancy in the L_p-metric, $2 \leqslant p \leqslant \infty$, *Dokl. Akad. Nauk SSSR*, **252** (1980), 805–807. [Russian] [English transl. *Soviet. Math. Dokl.* **21** (1980), 840–842]
7. Halton, J. H., On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals, *Numer. Math.*, **2** (1960), 84–90.
8. Hammersley, J. M., Monte Carlo methods for solving multivariable problems, *Ann. New York Acad. Sci.* **86** (1960), 844–874.
9. Heinrich, S., Efficient algorithms for computing the L_2 discrepancy, *Math. Comput.* **65** (1996), 1621–1633.
10. Hickernell, F. J., The mean square discrepancy of randomized nets, *ACM Trans. Modeling Comput. Simul.* **6**, No. 4 (1996), 274–296.
11. James, F., Hoogland, J., and Kleiss, R., Multidimensional sampling for simulation and integration: Measures, discrepancies, and quasi-random numbers, *Comput. Phys. Commun.* **99** (1997), 180–220.
12. Kuipers, L., and Niederreiter, H., "Uniform Distribution of Sequences," Wiley, New York, 1974.
13. Luby, M., and Wigderson, A., "Pairwise independence and derandomization," Tech. Report UCB/CSD-95-880, Univ. of California at Berkeley, 1995. [Available electronically at `http://www.icsi.berkeley.edu/~luby/pair_sur.html`]
14. Montgomery, H. L., "Ten Lectures on the Interface between Analytic Number Theory and Harmonic Analysis," CBMS Regional Conference Series in Mathematics, No. 84, Math. Soc., Providence, RI, 1994.
15. Morokoff, W., and Caflisch, R., Quasi-random sequences and their discrepancies, *SIAM J. Sci. Comput.* **15** (1994), 1251–1279.
16. Morokoff, W., and Caflisch, R., Quasi-Monte Carlo integration, *J. Comput. Phys.* **122** (1999), 218–230.
17. Motwani, R., and Raghavan, P., "Randomized Algorithms," Cambridge Univ. Press, Cambridge, 1995.
18. Niederreiter, H., "Random Number Generation and Quasi-Monte Carlo Methods," CBMS-NSF Regional Conference Series in Applied Mathematics, Vol. 63, SIAM, Philadelphia, 1992.
19. Niederreiter, H., and Xing, C. P., Low-discrepancy sequences and global function fields with many rational places, *Finite Fields Appl.* **2** (1996), 241–273.
20. Ninomiya, S., and Tezuka, S., Towards real-time pricing of complex financial derivatives, *Appl. Math. Finance* **3** (1996), 1–20.
21. Owen, A. B., Monte-Carlo variance of scrambled net quadrature, *SIAM J. Numer. Anal.* **34**, No. 5 (1997), 1884–1910.
22. Paskov, S. H., and Traub, J. R., Faster valuation of financial derivatives, *J. Portfolio Manage.* (1995), 113–120.
23. Richtmyer, R. D., "The Evaluation of Definite Integrals, and Quasi-Monte Carlo Method Based on the Properties of Algebraic Numbers.," Report LA-1342, Los Alamos Scientific Laboratory, Los Alamos, NM, 1951.
24. Roth, K. F., On irregularities of distribution, *Mathematika* **1** (1954), 73–79.
25. Roth, K. F., On irregularities of distribution IV, *Acta Arith.* **37** (1996), 67–75.
26. Skriganov, M. M., Constructions of uniform distributions in terms of geometry of numbers, *Algebra Anal.*, **6** (1994), 200–223. [*St. Petersburg Math. J.* **6** (1995), 635–664]

27. Sloan, I. H., and Woźniakowski, H., An intractability result for multiple integration, *Math. Comput.* **66** (1997), 1119–1124.

28. Spanier, J., and Maize, E. H., Quasi-random methods for estimating integrals using relatively small samples, *SIAM Rev.* **36** (1994), 18–44.

29. Tezuka, S., "Uniform Random Numbers. Theory and Practice," Kluwer Academic, Dordrecht, 1995.

30. Janse van Rensburg, E. J., and Torrie, G. M., Estimation of multidimensional integrals: Is Monte Carlo the best method?, *J. Phys. A: Math. Gen.* **26** (1993), 943–953.

31. Warnock, T. T., Computational investigations of low-discrepancy point sets, *in* "Applications of Number Theory to Numerical Analysis" (S. K. Zaremba, Ed.), pp. 319–343, Academic Press, New York, 1972.