Interner Bericht

Efficient algorithms for computing the L_2 discrepancy

Stefan Heinrich

267/95

Fachbereich Informatik

Universität Kaiserslautern · Postfach 3049 · D-67653 Kaiserslautern

Efficient algorithms for computing the L_2 discrepancy

Stefan Heinrich

(submitted to: Mathematics of Computation)

267/95

Universität Kaiserslautern
AG Numerische Algorithmen
Postfach 30 49
67653 Kaiserslautern

März 1995

Herausgeber: AG Numerische Algorithmen
Leiter: Professor Dr. S. Heinrich

Efficient algorithms for computing the L_2 discrepancy

S. Heinrich
Fachbereich Informatik
Universität Kaiserslautern
67653 Kaiserslautern
e-mail: heinrich@informatik.uni-kl.de

Abstract

The L_2-discrepancy is a quantitative measure of precision for multivariate quadrature rules. It can be computed explicitly. Previously known algorithms needed $O(m^2)$ operations, where m is the number of nodes. In this paper we present algorithms which require $O(m(\log m)^d)$ operations.

1 Introduction

Let $A = ((x_1, v_1), \ldots, (x_m, v_m))$ be an array defining a quadrature formula on $G = [0,1]^d$, i.e. $x_i \in G$, $v_i \in \mathbb{R}$ $(i = 1, \ldots, m)$, and the quadrature is given by

$$Qf = \sum_{i=1}^{m} v_i f(x_i)$$

for any continuous function $f \in C(G)$. Given $t = (t_1, \ldots, t_d) \in G$, we let

$$e(t) = \int_G \chi_{[0,t)}(x)dx - \sum_{i=1}^{m} v_i \chi_{[0,t)}(x_i),$$

where $[0,t) = \prod_{k=1}^{d} [0, t_k)$. If $v_i = 1/m$ for all i, then $e(t)$ measures the local deviation of the empirical distribution of the point set $\{x_i : i = 1, \ldots, m\}$ from the uniform distribution:

$$e(t) = \text{mes}\,([0,t)) - \frac{|\{i : x_i \in [0,t)\}|}{m}.$$

Here $|X|$ denotes the cardinality of a set X. The L_2-discrepancy of A is defined by

$$D_2(A) = \left(\int_G e(t)^2 dt \right)^{1/2}.$$

So $D_2(A)$ is the mean square error of the quadrature A, applied to characteristic functions $\chi_{[0,t)}$. Besides this meaning the L_2-discrepancy possesses further general interpretations. Denote by BW_2^1 the set of all functions $f \in L_2(G)$ whose generalized mixed derivative satisfies

$$\frac{\partial^r f(s)}{\partial s_1 \dots \partial s_d} \in L_2(G)$$

$$\|\frac{\partial^r f(s)}{\partial s_1 \dots \partial s_d}\|_{L_2(G)} \le 1$$

and $f(s) = 0$ whenever $s_k = 1$ for some $k = 1, \dots, d$. Then

$$D_2(A) = \sup_{f \in BW_2^1} |If - Qf|,$$

where If stands for the integral $\int_G f(x)dx$. So $D_2(A)$ is the worst-case error over a Sobolev class with bounded mixed derivative (see e.g. [Tem90]). As recently proved by Woźniakowski [Woź91], the L_2 discrepancy also appears as an average case error with respect to the Wiener sheet measure: So let μ denote the mean zero Gaussian measure on $C(G)$ given by the covariance kernel

$$R_\mu(s,t) = \prod_{k=1}^{d} \min(s_k, t_k).$$

Then

$$D_2(\widetilde{A}) = \left(\int_G |If - Qf|^2 d\mu(f)\right)^{1/2},$$

where $\widetilde{A}$ is obtained from A by replacing x_i by $\widetilde{x}_i = \bar{1} - x_i$, and $\bar{1} = (1, 1, \dots, 1)$.
The L_2-disrepancy was studied in many papers. We refer to the surveys [Nie78], [Nie92] as well as to [Tem90]. The order of the smallest possible discrepancy was determined in [Rot54], [Dav56], [Rot80], [Fro80], [Byk85]:

$$\inf\{D_2(A) : \text{All possible } A \text{ with } m \text{ nodes}\} = \Theta(m^{-1}(\log m)^{(d-1)/2})$$

By now several ways are known of constructing quadratures which attain this optimal rate or attain it up to powers of $\log n$ (see again the references above). So the asymptotic order cannot distinguish between such quadratures, and one also wants to have precise numerical information. By integrating over $t \in [0,1]^d$ it is not difficult to derive the following explicit formula for the square of the discrepancy.

$$\begin{aligned} D_2(A)^2 &= \int_G \left(\prod_{k=1}^{d} t_k\right)^2 dt - 2\sum_{i=1}^{m} v_i \int_G \prod_{k=1}^{d} t_k \chi_{[0,t_k)}(x_{ik}) dt \\ &\quad + \sum_{i,j=1}^{m} v_i v_j \int_G \prod_{k=1}^{d} \chi_{[0,t_k)}(x_{ik}) \chi_{[0,t_k)}(x_{jk}) dt \\ &= 3^{-d} - 2^{1-d} \sum_{i=1}^{m} v_i \prod_{k=1}^{d} (1 - x_{ik}^2) + \sum_{i,j=1}^{m} v_i v_j \prod_{k=1}^{d} (1 - \max(x_{ik}, x_{jk})). \qquad (1) \end{aligned}$$

This formula was first pointed out and used for the numerical investigation of various low discrepancy sets by Warnock [War72]. Since then, many experimental investigations were based on it.

The second term of (1) is computed in $O(m)$ operations, the straightforward computation of the third term requires $O(m^2)$ operations (by operation we mean either an arithmetic operation or a comparison). This makes the computation of $D_2(A)$ for large A a highly complex task. So far no algorithms were known of lower complexity. (Note that there were recent efforts to design efficient algorithms for another type of discrepancy - the star-discrepancy, which is obtained by taking the L_∞ norm of $e(t)$ instead of the L_2 norm. See [DE93], [DG94]). The aim of the present paper is to give an algorithm which is of worst case complexity $O(m(\log m)^d)$, and an easier to implement modification of it which has average case complexity $O(m(\log m)^d)$.

2 The algorithm and its worst-case analysis

We shall present an algorithm which computes the third term of (1) in $O(m(\log m)^d)$ operations. The algorithm D is defined recursively and will accomplish a slightly more general task. Given another array $B = ((y_1, w_1), \ldots, (y_n, w_n))$ with $y_j \in G,\ w_j \in \mathbb{R}$ $(j = 1, \ldots, n)$, the algorithm will compute

$$D(A, B, d) = \sum_{i=1}^{m} \sum_{j=1}^{n} v_i w_j \prod_{k=1}^{d} (1 - \max(x_{ik}, y_{jk})). \tag{2}$$

Before we give a more formal description of the algorithm let us first explain the basic idea. Suppose we know that the first coordinates of A are all not greater than those of B, i.e.

$$x_{i1} \leq y_{j1} \quad (i = 1, \ldots, m,\ j = 1, \ldots, n).$$

Then (2) simplifies to

$$D(A, B, d) = \sum_{i=1}^{m} \sum_{j=1}^{n} v'_i w'_j \prod_{k=2}^{d} (1 - \max(x_{ik}, y_{jk})),$$

where $v'_i = v_i$ and $w'_j = (1 - y_{j1}) w_j$. Hence we have reduced the dimension of the problem by 1. Suppose now that $d = 1$. Then after the reduction we are left with the double sum

$$\sum_{i=1}^{m} \sum_{j=1}^{n} v'_i w'_j = \left(\sum_{i=1}^{m} v'_i \right) \left(\sum_{j=1}^{n} w'_j \right)$$

which now can be computed in $O(m + n)$ operations. The algorithms is recursive and applies the divide-and-conquer strategy to reduce the dimension. Let us pass to the details.

We assume that A is sorted in such a way that

$$x_{11} \leq x_{21} \leq \ldots \leq x_{m1}. \tag{3}$$

This can be achieved by an initial sorting in $O(m \log m)$ time. (This initial sorting is not part of the algorithm D, but in the recursion the algorithm will take care of such an ordering itself).
We formally also include the case $B = \emptyset$ and the case $d = 0$. In the latter we suppose $A = (v_1, \ldots, v_m)$ and $B = (w_1, \ldots, w_n)$.

Algorithm D

Input: A, B, d as above, A satisfying $A \neq \emptyset$ and (3).

Output: $D(A, B, d)$

Case 1: $n = 0$ (i. e. $B = \emptyset$)

$$D(A, B, d) = 0$$

Case 2: $d = 0$, $n \geq 1$

$$D(A, B, 0) = \left(\sum_{i=1}^{m} v_i\right)\left(\sum_{j=1}^{n} w_j\right)$$

Case 3: $m = 1$, $d \geq 1$, $n \geq 1$

$$D(A, B, d) = v_1 \sum_{j=1}^{n} w_j \prod_{k=1}^{d} (1 - \max(x_{1k}, y_{jk}))$$

Case 4: $m > 1$, $d \geq 1$, $n \geq 1$
Set $p = [\frac{m}{2}]$, $\xi = x_{p1}$. Form new arrays A_1, A_2, B_1, B_2 as follows:

$$A_1 = ((x_1, v_1), \ldots, (x_p, v_p)) \quad (4)$$
$$A_2 = ((x_{p+1}, v_{p+1}), \ldots, (x_m, v_m)). \quad (5)$$

To define the arrays B_1, B_2, we treat the elements of B consecutively. All elements whose first coordinate is not greater than ξ go into B_1, the rest goes into B_2. Precisely, we put

$$B_1 = ((y_{j_1}, w_{j_1}), \ldots, (y_{j_q}, w_{j_q})) \quad (6)$$
$$B_2 = ((y_{j_{q+1}}, w_{j_{q+1}}), \ldots, (y_{j_n}, w_{j_n}), \quad (7)$$

where q and the j_k are defined through the relations

$$y_{j_k,1} \leq \xi \quad (k = 1, \ldots, q)$$
$$y_{j_k,1} > \xi \quad (k = q+1, \ldots, n)$$

and

$$j_1 < j_2 < \ldots < j_q,$$
$$j_{q+1} < j_{q+2} < \ldots < j_n.$$

Let P' be the projection of $\mathbb{R}^d$ onto $\mathbb{R}^{d-1}$ given by omitting the first coordinate. Put

$$\begin{array}{lcll} x'_i & = & P'x_i & (i=1,\ldots,m) \\ y'_j & = & P'y_j & (j=1,\ldots,n) \\ v'_i & = & v_i & (i=1,\ldots,p) \\ v'_i & = & v_i(1-x_{i1}) & (i=p+1\ldots,m) \\ w'_{j_k} & = & w_{j_k} & (k=1,\ldots,q) \\ w'_{j_k} & = & w_{j_k}(1-y_{j_k1}) & (k=q+1,\ldots,n). \end{array}$$

Form the sets A'_1, A'_2, B'_1, B'_2 defined by literally putting primes to the symbols in (4) - (7). Obtain A''_1, A''_2 from A'_1, A'_2 by sorting with respect to the (new) first coordinate, so that (3) holds for these new arrays. In the case $d=1$ the definitions above have to be interpreted in the appropriate way: the primed arrays consist only of the v'_i and w'_j. In this case the sorting step is omitted, so $A''_1 = A'_1, A''_2 = A'_2$.

Finally, we set

$$D(A,B,d) \;=\; D(A_1,B_1,d) + D(A_2,B_2,d) + D(A''_1,B'_2,d-1) + D(A''_2,B'_1,d-1).$$

This recursion completes case 4 and the algorithm.

It is readily checked that

$$\begin{array}{rcl} D(A''_1,B'_2,d-1) & = & D(A_1,B_2,d) \\ D(A''_2,B'_1,d-1) & = & D(A_2,B_1,d), \end{array}$$

and hence the algorithm indeed computes the desired quantity (2).

Let us estimate the maximal number of operations $L(m,n,d)$ over all possible inputs of size m,n and dimension d. Let us assume that we use a sorting algorithm with (worst-case) number of operations at most $c_{sort} n \log n$ (the logarithm to the base 2), with some constant $c_{sort} > 0$.

Proposition 1 *For each $d \geq 0$ there exists a constant $c_d > 0$ such that for all $m \geq 1, n \geq 0$,*

$$L(m,n,d) \leq c_d(m+n)(\log m + 1)^d. \tag{8}$$

Proof: For $n=0$ we have $L(m,0,d)=0$, and (8) holds trivially. For $d=0, n\geq 1$ we are in case 2 and have

$$L(m,n,0) = m+n-1,$$

so we put $c_0 = 1$. For $d \geq 1$ we define

$$c_d = \max(3d+1, (\log 1.5)^{-1}(c_{d-1} + c_{sort} + 3)). \tag{9}$$

By induction over m we shall prove that (8) holds for all $d \geq 1, n \geq 1$. For $m=1$, which is case 3, it follows that

$$L(1,n,d) = (3d+1)n \leq c_d(n+1).$$

Now we fix $m > 1$, put $p = [\frac{m}{2}]$ and $\sigma(d) = 1$ if $d > 1$, $\sigma(d) = 0$ if $d = 1$. From case 4 we deduce

$$\begin{aligned} L(m,n,d) \leq & \max_{0\leq q\leq n} \{n + 2(m-p+n-q) + \sigma(d)c_{sort}(p\log p + (m-p)\log(m-p)) \\ & + L(p,q,d) + L(m-p,n-q,d) \\ & + L(p,n-q,d-1) + L(m-p,q,d-1) + 3\}. \end{aligned} \tag{10}$$

Observe that , since $m > 1$,

$$\max(p, m-p) = \left[\frac{m+1}{2}\right] \leq 2m/3.$$

Using this and the induction hypothesis, we obtain,

$$\begin{aligned} L(m,n,d) \leq & \max_{0\leq q\leq n} \{3(m+n) + \sigma(d)c_{sort}m(\log m + 1) \\ & + c_d(p+q)(\log p + 1)^d + c_d(m-p+n-q)(\log(m-p)+1)^d \\ & + c_{d-1}(p+n-q)(\log p + 1)^{d-1} + c_{d-1}(m-p+q)(\log(m-p)+1)^{d-1}\} \\ \leq & (3 + c_{sort})(m+n)(\log m + 1)^{d-1} + c_d(m+n)(\log(2m/3)+1)^d \\ & + c_{d-1}(m+n)(\log m + 1)^{d-1}. \end{aligned}$$

Since

$$\begin{aligned} (\log(2m/3)+1)^d &= (\log m + 1 - \log 1.5)^d \\ &\leq (\log m + 1 - \log 1.5)(\log m + 1)^{d-1} \\ &= (\log m + 1)^d - \log 1.5(\log m + 1)^{d-1}, \end{aligned} \tag{11}$$

we conclude

$$\begin{aligned} L(m,n,d) \leq & (3 + c_{sort} + c_{d-1})(m+n)(\log m + 1)^{d-1} \\ & + c_d(m+n)(\log m + 1)^d - c_d \log 1.5(m+n)(\log m + 1)^{d-1} \\ \leq & c_d(m+n)(\log m + 1)^d. \end{aligned}$$

Clearly, the constants are overestimated by (9) - for the sake of convenience in the proof. Tight upper bounds could be calculated numerically on the basis of (10). Having algorithm D, it is clear how to compute $D_2(A)$: We determine the first two terms of (1), then we sort A so that it has nondecreasing first coordinates, and finally we apply $D(A, A)$. Clearly, this takes not more than $O(m(\log m)^d)$ operations.

3 A modification efficient on the average

The second algorithm is a simplification of D in that it avoids the sorting. In multivariate integration one studies quadrature formulas whose nodes are as close to the equidistribution as possible. So looking at the first coordinate, one should expect that for about one half of the nodes it is below $1/2$. This is exploited in algorithm D', which is close to D, but sets the (initial) ξ to $1/2$. It seems that algorithm D' is better suited for practical purposes. Of course "bad" node sets can spoil the performance of D', but we shall prove later that on the average it still finishes after $O(m(\log m)^d)$ operations.

Algorithm D'

Input: A, B, d as above (A needs not to be sorted), reals $a, b \in [0,1], a < b$, and we assume that $x_{i1}, y_{j1} \in [a,b]$ $(i = 1,\ldots,m,\ j = 1,\ldots,n)$.

Output: $D'(A,B,d,a,b) = D(A,B,d)$

The cases $d = 0$, $m = 0$ or $n = 0$ are handled in analogy with algorithm D, so we restrict our description to the essential case:

Recursion: Assume $d \geq 1, m, n \geq 1$.

Set $\xi = (a+b)/2$ and put an element of A into A_1, if its first coordinate does not exceed ξ, otherwise into A_2. Form B_1 and B_2 in the same way. If $\max(|A_1|, |A_2|) > 2m/3$, then we compute $D(A,B,d)$ directly by (2) (that means in $O(mn)$ operations). Otherwise we define A_1', A_2', B_1', B_2' as in algorithm D and set

$$\begin{aligned} D'(A,B,d,a,b) &= D'(A_1,B_1,d,a,\xi) + D'(A_2,B_2,d,\xi,b) + D'(A_1',B_2',d-1,0,1) \\ &\quad + D'(A_2',B_1',d-1,0,1). \end{aligned} \tag{12}$$

Now we shall study the average behaviour of D'. First we consider the case of independent sets A and B, the case $A = B$ is treated afterwards. So let us assume that x_i $(i = 1,\ldots,m)$ and y_j $(j = 1,\ldots,n)$ are independent random variables, uniformly distributed in $[a,b] \times [0,1]^{d-1}$. Let $E(m,n,d)$ be the expected number of operations of algorithm D' (it is obvious that this number does not depend on a and b).

Proposition 2: For each $d \geq 0$ there exists a constant $c_d' > 0$ such that for all $m \geq 1,\ n \geq 0$

$$E(m,n,d) \leq c_d'(m+n)(\log m + 1)^d. \tag{13}$$

Proof: The case $d = 0$ is treated as above. Let $m_0 \in \mathbb{N}$ be a constant, fixed in such a way that $2m \exp(-m/18) \leq 1$ whenever $m > m_0$. Clearly, for $m \leq m_0$, algorithm D' needs $O(n)$ operations, so an appropriate choice of c_d' ensures (13) for all $m \leq m_0$ and all n. Now we shall proceed be induction over m and assume that $m > m_0$.

We make use of a special case of Chernoff's technique, see e. g. [Mul, Th. A.1.1] (also called Kolmogorov-Bernstein-inequality). For a sequence $\eta_1, \ldots, \eta_m$ of independent random variables, each taking the values 1 and -1 with probability 1/2, we have for all $\varepsilon > 0$

$$\mathrm{Prob}\left\{\sum_{i=1}^{m} \eta_i > \varepsilon m\right\} \leq \exp(-\varepsilon^2 m/2).$$

Setting $\eta_i = 1$ if $x_{i1} \leq \xi$ and $\eta_i = -1$ otherwise, we obtain with $\varepsilon = 1/3$

$$\mathrm{Prob}\{|A_1| > 2m/3\} \leq \exp(-m/18).$$

By symmetry

$$\mathrm{Prob}\ \{|A_2| > 2m/3\} \leq \exp(-m/18),$$

so we get

$$\text{Prob}\,\{\max(|A_1|, |A_2|) > 2m/3\} \leq 2\,\exp(-m/18). \tag{14}$$

Next let us fix some further notation.
Set

$$\begin{aligned}\mu_\ell &= \text{Prob }\{|A_1| = \ell\}\,,\\ \nu_\ell &= \text{Prob }\{|B_1| = \ell\}\end{aligned}$$

and

$$S = \{(p,q) : p \in \{0,1,\ldots,m\}\ ,\ \max(p, m-p) \leq 2m/3,\ q \in \{0,1,\ldots,n\}\}$$

$$T = \{0,1,\ldots,m\} \times \{0,1,\ldots,n\} \backslash S.$$

Then (14) gives

$$\sum_{(p,q)\in T} \mu_p \nu_q = \sum_{\max(p,m-p)>2m/3} \mu_p \sum_{q=0}^{n} \nu_q \leq 2\,\exp(-m/18). \tag{15}$$

Let us now consider (12). Fix $p \in \{0,1,\ldots,m\}$. Under the condition that $|A_1| = p$, the conditional distribution of those x_{i1} which fall into $[a,\xi]$ is that of a sequence of p independent equidistributed over $[a,\xi]$ random variables. An analogous relation holds for $q \in \{0,1,\ldots,n\}$ and $|B_1| = q$.
So under the condition $|A_1| = p$ and $|B_1| = q$ the expected number of operations to accomplish $D'(A_1, B_1, d, a, \xi)$ is just $E(p,q,d)$. Similar remarks apply to the remaining three parts of the recursion (12). Note further that the recursion switches to the direct computation if and only if $(p,q) \in T$. Summing first over j, then over i, this direct evaluation can be accomplished in $m(3d+1)n + m - 1$ operations. Summarizing this, we get

$$\begin{aligned}E(m,n,d) &= \sum_{(p,q)\in S} \mu_p\nu_q\{m + n + 2(m-p+n-q) + E(p,q,d) + E(m-p, n-q, d)\\ &\quad + E(p, n-q, d-1) + E(m-p, q, d-1) + 3\}\\ &\quad + (m(3d+1)n + m - 1) \sum_{(p,q)\in T} \mu_p\nu_q.\end{aligned}$$

The induction hypothesis and (15) imply

$$\begin{aligned}E(m,n,d) &\leq \sum_{(p,q)\in S} \mu_p\nu_q\{4(m+n) + c'_d(p+q)(\log p + 1)^d\\ &\quad + c'_d(m-p+n-q)(\log(m-p)+1)^d + c'_{d-1}(p+n-q)(\log p + 1)^{d-1}\\ &\quad + c'_{d-1}(m-p+q)(\log(m-p)+1)^{d-1}\Big\} + 2\,\exp(-m/18)m((3d+1)n+1).\end{aligned}$$

Since $m > m_0$, the last term can be estimated by $(3d+1)(m+n)$, according to the choice of m_0 at the beginning of the proof. For $(p,q) \in S$ we have $\max(p, m-p) \leq 2m/3$, hence

$$E(m,n,d) \leq (3d+5)(m+n) + c'_d(m+n)(\log(2m/3)+1)^d + c'_{d-1}(m+n)(\log m + 1)^{d-1}.$$

Using (11), we get

$$E(m,n,d) \le c'_d(m+n)(\log m+1)^d + (3d+5+c'_{d-1} - c'_d \log 1.5)(m+n)(\log m+1)^{d-1}$$

To complete the induction, we arrange c'_d in such a way that the second term is not positive.

Now we turn to the case $B = A$. Here the recursion (12) can be simplified, since

$$D'(A'_1, A'_2, d-1, 0, 1) = D'(A'_2, A'_1, d-1, 0, 1).$$

Note that for $B = A$ the direct computation (2) can also be arranged in a more economical way:

$$\begin{aligned} &\sum_{i,j=1}^{m} v_i v_j \prod_{k=1}^{d} (1 - \max(x_{ik}, x_{jk})) \\ = &\sum_{i=1}^{m} v_i^2 \prod_{k=1}^{d} (1 - x_{ik}) + 2\sum_{i=1}^{m} v_i \sum_{j=i+1}^{m} v_j \prod_{k=1}^{d} (1 - \max(x_{ik}, x_{jk})), \end{aligned}$$

requiring a total of

$$\frac{3d+1}{2} m(m-1) + (2d+3)m$$

operations. Now we assume that $x_1, \ldots, x_m$ are drawn independently and uniformly distributed over G and denote the expected number of operations for $D'(A, A, d, a, b)$ by $E^*(m,d)$.

Proposition 3 For each $d \ge 0$ there exists a constant $c^*_d > 0$ such that for all $m \ge 1$

$$E^*(m,d) \le c^*_d m(\log m + 1)^d.$$

Proof: We start with the following observation: Fix p and indices $i_1 < i_2 < \ldots < i_p$, $i_{p+1} < i_{p+2} < \ldots < i_m$, with

$$\{i_1, \ldots, i_m\} = \{1, \ldots, m\}$$

(as sets). Under the condition that $x_{i_k 1} \in [0, 1/2]$ $(k = 1, \ldots, p)$ and $x_{i_k 1} \in \left(\frac{1}{2}, 1\right]$ $(k = p+1, \ldots, m)$, the random variables $x_{i_1}, \ldots, x_{i_p}$ and $x_{i_{p+1}}, \ldots, x_{i_m}$ are independent and equidistributed in $[0, \frac{1}{2}] \times [0,1]^{d-1}$ and $[\frac{1}{2}, 1] \times [0,1]^{d-1}$ respectively. From this observation one easily derives

$$\begin{aligned} E^*(m,d) \quad = \quad &\sum_{\max(p,m-p) \le 2m/3} \mu_p \{m + m - p + E^*(p,d) + E^*(m-p,d) \\ &+ E(p, m-p, d-1) + 3\} + \left(\frac{3d+1}{2} m(m-1) + (2d+3)m\right) \sum_{\max(p,m-p) > 2m/3} \mu_p. \end{aligned}$$

Using Proposition 2 and relation (14), an inductive argument analogous to the previous one completes the proof.

4 Generalizations

Instead of rectangles with lower left corner fixed at the origin one may consider all rectangles in G with sides parallel to the axes. This leads to the so-called unanchored L_2-discrepancy, defined as follows: Put

$$e(s,t) = \int_G \chi_{[s,t)}(x)dx - \sum_{i=1}^{m} v_i \chi_{[s,t)}(x_i)$$

whenever $s_k \le t_k$ for all $k = 1, \ldots, d$, and

$$\Delta_2(A) = \left(\int_G \int_{[0,t]} e(s,t)^2 ds dt \right)^{1/2}.$$

Calculating the integrals gives

$$\Delta_2(A)^2 = 12^{-d} - 2 \cdot 6^{-d} \sum_{i=1}^{m} v_i \prod_{k=1}^{d} (1 - x_{ik}^3 - (1 - x_{ik})^3)$$
$$+ \sum_{i,j=1}^{m} v_i v_j \prod_{k=1}^{d} \min(x_{ik}, x_{jk})(1 - \max(x_{ik}, x_{jk})),$$

and it is clear that algorithms D and D' have immediate extensions to this case of about the same efficiency. Another generalization is that to higher smoothness. Instead of testing the quadrature on characteristic functions $\chi_{[0,t)}(x)$ we fix an integer $r > 0$ and test on

$$B_r(x,t) = (r!)^{-d} \prod_{k=1}^{d} (t_k - x_k)_+^r,$$

where $(t_k - x_k)_+$ stands for $(t_k - x_k)$ if $t_k \ge x_k$ and for 0 otherwise. Setting

$$e_r(t) = \int_G B_r(x,t)dx - \sum_{i=1}^{m} v_i B_r(x_i,t)$$

and

$$D_2^{(r)}(A) = \left(\int_G e_r(t)^2 dt \right)^{1/2}, \tag{16}$$

we obtain an r-smooth analogue of the discrepancy, which was considered in [Tem90], [Pas93]. $D_2^{(r)}(A)$ can be interpreted again as a worst case error of Q over a certain Sobolev class of functions $f \in L_2(G)$ which satisfy

$$\left\| \frac{\partial^{(r+1)d} f(s)}{\partial s_1^{r+1} \ldots \partial s_d^{r+1}} \right\|_{L_2(G)} \le 1,$$

and certain boundary conditions (see [Tem90], [Pas93]). The analogy to $r = 0$ extends also to the average case. $D_2^{(r)}(\tilde{A})$ can be shown to be equal to the average error of Q with respect to a certain "r-smooth" Wiener measure. For details we refer to [Pas93]. Expanding the integral in relation (16), we get after elementary calculations

$$\begin{aligned} D_2^{(r)}(A)^2 &= ((r+1)!)^{-2d}(2r+3)^{-d} - 2(r!(r+1)!)^{-d}\sum_{i=1}^{m} v_i \prod_{k=1}^{d}\int_{x_{ik}}^{1} \tau^{r+1}(\tau - x_{ik})^r d\tau \\ &+ \sum_{i,j=1}^{m} v_i v_j \prod_{k=1}^{d} \varphi(x_{ik}, x_{jk}), \end{aligned}$$

where

$$\varphi(a,b) = (r!)^{-2}\int_{\max(a,b)}^{1} (\tau - a)^r(\tau - b)^r d\tau \tag{17}$$

for $a, b \in [0,1]$. Proceeding as in the case $r = 0$ we seek to compute

$$D^{(r)}(A,B) = \sum_{i=1}^{m}\sum_{j=1}^{n} v_i w_j \prod_{k=1}^{d} \varphi(x_{ik}, y_{jk}).$$

Assume now

$$x_{i1} \le y_{j1} \quad (i = 1, \ldots, m,\ j = 1, \ldots, n).$$

Then the dimension reduction can be done in the following way: It is easily checked that for $a \le b$ the function $\varphi(a,b)$ can be represented as

$$\varphi(a,b) = \sum_{\ell=0}^{r} a^\ell p_\ell(b) \tag{18}$$

with certain polynomials p_ℓ of degree not exceeding $2r+1-\ell$. Hence

$$D^{(r)}(A,B) = \sum_{\ell=0}^{r}\sum_{i=1}^{m}\sum_{j=1}^{n} v_i^{(\ell)} w_j^{(\ell)} \prod_{k=2}^{d} \varphi(x_{ik}, y_{jk})$$

with

$$\begin{aligned} v_i^{(\ell)} &= v_i x_{i1}^\ell \\ w_j^{(\ell)} &= w_j p_\ell(y_{j1}). \end{aligned}$$

So we are left with $r+1$ problems of dimension $d-1$. On this basis one can show again that $D_2^{(r)}(A,B)$ can be computed in $O((m+n)(\log m)^d)$ operations. This time, however, each dimension reduction multiplies the effort by $(r+1)$, so a heavy dependence of the constants on the dimension can be expected, which still makes the $O(mn)$ algorithm preferable except for small r and small d.

5 Numerical experiments

Here we present the results of a few first tests which were carried out for algorithm D'. The aim was to understand the speed-up, so we didn't list the discrepancies, but the number of operations of the recursive algorithm. This figure depends not only on m and d, but also on the concrete sequence considered, and was determined in the process of computation. The number of operations of the direct algorithm can be calculated as function of m and d only, by the formula $(3d+1)m(m-1)/2+(2d+3)m$, mentioned in section 3. The nodes were formed by the Halton sequence with $m = 1024$, $m = 8192$ and $m = 65536$, respectively. We tested dimensions $d = 1, 2, 4, 6$ and 8. The program was a first implementation of algorithm D', so none of the possible optimizations discussed in section 6 below had yet been tried. Consequently these experiments were rather meant to give a first impression than a conclusive picture. More detailed findings will be reported elsewhere.

number of points m	dimension d	number of operations of direct computation	number of operations of algorithm D'
1024	1	2.1 E 6	6.8 E 4
	2	3.7 E 6	3.3 E 5
	4	6.8 E 6	2.4 E 6
	6	1.0 E 7	6.3 E 6
	8	1.3 E 7	1.0 E 7
8192	1	1.3 E 8	7.0 E 5
	2	2.3 E 8	4.3 E 6
	4	4.4 E 8	5.3 E 7
	6	6.4 E 8	2.2 E 8
	8	8.4 E 8	4.6 E 8
65536	1	8.6 E 9	6.8 E 6
	2	1.5 E 10	5.2 E 7
	4	2.8 E 10	9.5 E 8
	6	4.1 E 10	5.9 E 9
	8	5.4 E 10	1.8 E 10

The calculations were done on a HP 9000/735 workstation.

6 Remarks and open problems

It is seen from the experiments and to be expected from the theoretical analysis that the recursive algorithms are particularly advantageous in low dimension. In high dimension only little is gained, and storage management, which is not reflected in the table above, may become an additional problem. So efficient data structures should be an issue of further investigation.

Both algorithms leave enough space for various speed-up strategies. For example, both methods could switch from recursion to the direct computation if the number of elements in the actual A gets smaller than a certain threshold. A possible candidate could be 2^d, where d is the actual dimension, since with less elements in A the recursion hardly gets down to dimension 0 before the sets reach cardinality 0 or 1. The switch to direct computation in algorithm D' is sufficient for the theoretical average analysis, but could be modified for practical purposes. Even if A_1 is large as compared to A_2 it might be advantageous to divide

A_1 further. Other strategies for the choice of ξ could be imagined, as well, for example the arithmetic mean of the first coordinates of A. The $O(m \log m)$ sorting of algorithm D could be replaced by an order statistics procedure which determines the $[\frac{m}{2}]$-th smallest element ξ of A in $O(m)$ operations (see [AHU74], ch. 3) and then produces A_1 and A_2 as in algorithm D'. This might improve the practical behaviour, the power of logarithm in the total cost estimate will not be decreased, however.
The case $r \geq 1$ leaves open even more questions. In each step a large number of subproblems arises, so the total effort, as compared to $r = 0$, gets multiplied by a factor exponential in d. More efficient ways of handling the integral (17) are needed, e.g. more economical splittings (18) of the function φ into products of polynomials of one variable.
Finally, there arises an interesting problem in algebraic complexity. Is $O(m(\log m)^d)$ also a lower bound for any algorithm computing the L_2-discrepancy?

Acknowledgement

I thank D. Lee and K. Frank for helpful comments and M. Taubert for the implementation of the algorithms.

References

[AHU74] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Massachusetts, 1974.

[Byk85] V. A. Bykovskii. On the correct order of the error of optimal cubature formulas in spaces with dominant derivative, and on quadratic deviations of grids. Preprint, Computing Center Far-Eastern Scientific Center, Acad. Sci. USSR, Vladivostok, 1985. (Russian) R. Zh. Mat. 1986, 7B 1663.

[Dav56] H. Davenport. Note on irregularities of distribution. *Mathematika*, **3**: 131 – 135, 1956.

[DE93] D. Dobkin and D. Eppstein. Computing the discrepancy. *Proceedings of the Ninth Annual Symposium on Computational Geometry*, pages 47 – 52, 1993.

[DG94] D. Dobkin and D. Gunopulos. Computing the rectangle discrepancy. Preprint, Princeton University, 1994.

[Fro80] K. K. Frolov. An upper estimate of the discrepancy in the L_p-metric, $2 \leq p < \infty$. *Dokl. Akad. Nauk SSSR*, **252**: 805 – 807, 1980. English transl. in Soviet Math. Dokl. **21** (1980).

[Nie78] H. Niederreiter. Quasi-Monte Carlo methods and pseudo random numbers. *Bull. Amer. Math. Soc.*, **84**: 957 – 1041, 1978.

[Nie92] H. Niederreiter. Random Number Generation and Quasi-Monte Carlo Methods. *SIAM, Philadelphia*, 1992.

[Pas93] S. H. Paskov. Average complexity of multivariate integration for smooth functions. *J. Complexity*, **9**: 291 – 312, 1993.

[Rot54] K. F. Roth. On irregularities of distribution. *Mathematika*, **1**: 73 – 79, 1954.

[Rot80] K. F. Roth. On irregularities of distribution. *IV, Acta Arith.*, **37**: 67 – 75, 1980.

[Tem90] V. N. Temlyakov. On a way of obtaining lower estimates for the errors of quadrature formulas. *Mat. Sbornik*, **181**: 1403 – 1413, 1990. English transl.: Math. USSR Sbornik, **71**, (1992), pages 247 – 257.

[War72] T. T. Warnock. Computational investigations of low discrepancy point sets. In S. K. Zaremba, editor, *Applications of Number Theory to Numerical Analysis*. Academic Press, New York, 1972.

[Woź91] H. Woźniakowski. Average case complexity of multivariate integration. *Bull. Amer. Math. Soc.*, **24**: 185 – 194, 1991.