MATHEMATICS OF COMPUTATION
VOLUME 64, NUMBER 210
APRIL 1995, PAGES 807-818

ADAPTIVE SEARCH IN QUASI-MONTE-CARLO OPTIMIZATION

CHRISTIAN BIESTER, PETER J. GRABNER,
GERHARD LARCHER AND ROBERT F. TICHY

ABSTRACT. Motivated by a linear time-complexity result for an adaptive Monte Carlo algorithm, we propose and analyze an adaptive deterministic algorithm. We restrict a grid search to nested subregions that promise to provide improvement of the current solution, and we obtain an exponential rate of convergence in function evaluations. For proving the main result we restrict ourselves to functions on hypercubes. In a final section we outline how to extend the method to the general case and give some numerical examples.

1. INTRODUCTION

Let S be a closed and bounded subset of $\mathbb{R}^n$, f a continuous real-valued function defined on S, and $y^* := \max_{x\in S} f(x)$. In [8] a pure adaptive search algorithm for the approximate calculation of y^* is introduced provided that f is concave on a convex body S, and that $y^* = f(x)$ has a unique solution $x = x^*$.

Algorithm 1.

```
Step 0. Set $k = 0$, $S_0 = S$ and $y_0 \le \min_{x\in S} f(x) =: y_k$
Step 1. Generate $x_{k+1}$ uniformly distributed in $S_{k+1} := \{x | x \in S_k \text{ and } f(x) \ge y_k\}$
Step 2. Set $y_{k+1} = f(x_{k+1})$. If a (preset) stopping criterion is met, stop. Otherwise set $k := k + 1$ and return to Step 1.
```

By construction, y_k is an increasing sequence of points that almost surely converges to y^*. It is shown in [8] that for the expectation value $E(y_k)$ (in the standardized case $y_* := \min_{x\in S} f(x) = 0$, $y^* = 1$, $y_0 = y_*$)

$$E(y_k) \ge 1 - \left(\frac{n}{n+1}\right)^k . \tag{1.1}$$

Received by the editor July 6, 1993 and, in revised form, March 14, 1994.
1980 *Mathematics Subject Classification* (1985 *Revision*). Primary 65C05, 11K45, 65D99, 65K10.
The second and fourth authors are supported by the Austrian Science Foundation project Nr. P-10223-PHY.

Furthermore, the probabilistic estimate

$$\begin{aligned} k_{\alpha,m} &:= \min\left\{k \mid Pr\left(y_k \geq 1 - \frac{1}{m}\right) \geq 1 - \alpha\right\} \\ &\leq 2(n+1)\log\left(m\left(1 + \frac{1}{\sqrt{\alpha}}\right)\right) \end{aligned} \tag{1.2}$$

holds for all $m > 1$ and $0 < \alpha < 1$.

These results are shown by considering the worst case of this algorithm, i.e., taking $f = g$, where

$$g(x) := \inf\{y \mid (x, y) \in \text{ convex hull of } (x^*, y^*) \text{ and } (S, y_*)\}. \tag{1.3}$$

In applications of Algorithm 1 some striking problems occur: The authors of [8] write, "However, at the present time we do not have efficient procedures for generating uniformly distributed points in general convex regions." The actual and much more serious problem is that in Step 1 of Algorithm 1 it is assumed that the new set S_{k+1} is given explicitly (in a constructive way). But this, of course, in general is not the case, and so for practical purposes a further step would be necessary where S_{k+1} is determined or at least estimated. In this case it is doubtful whether the bound (1.2) remains linear in the dimension n or not. We remark here that in the forthcoming paper [10] linearity with respect to the dimension is extended to the case where f is 1-Lipschitz-continuous.

Another disadvantage (as is characteristic for all pure Monte Carlo methods) is, of course, that the results are merely probabilistic. The aim of the present paper is the presentation of an adaptive Quasi-Monte-Carlo algorithm which is now practicable in the form as stated. Furthermore, deterministic error estimates are established, and for a large class of functions exponential convergence to the maximum is shown.

2. The algorithm

Let for simplicity $S = [0, 1]^n$ be the n-dimensional unit cube, $f : S \mapsto \mathbb{R}$ be a function which is η-Lipschitz-continuous with $1 \leq \eta \leq 2$. This means that there exists a constant $L > 0$ such that for all $x_0 \in S$ there is an $A(x_0) \in \mathbb{R}^n$ satisfying

$$|f(x) - f(x_0) - A(x_0)(x - x_0)| \leq L\|x - x_0\|^\eta \tag{2.1}$$

for all $x \in S$. (By $\|\cdot\|$ we denote the maximum norm.)

Of course, if $\eta > 1$ then $A(x_0) = \operatorname{grad} f|x_0$ (by the definition of differentiability), and if $\eta = 1$ then we may take $A(x_0) = (0, \dots, 0)$.

Let $\max_{x \in S} f(x) =: y^*$ and $\varepsilon > 0$. We want to determine a point $\bar{x} \in S$ with $y^* - f(\bar{x}) \leq \varepsilon$.

Algorithm 2.

Step 0. $I_0 := S = [0, 1]^n$

Step 1. I_k is a union of cubes of the form

$$\prod_{i=1}^{n} \left[\frac{a_i}{2^k}, \frac{a_i + 1}{2^k}\right] \quad \text{with } 0 \leq a_i \leq 2^k.$$

Construct I_{k+1} in the following way: Divide each of the subcubes of I_k into 2^n subcubes of half side length in the natural way and set

$$M_{k+1} := \max\{f(x) \mid x \text{ is vertex of one of the new subcubes }\}.$$

Take I_{k+1} as the union of all new subcubes which have at least one vertex x with

$$f(x) \geq M_{k+1} - \frac{L}{2^{(k+1)\eta}}.$$

Step 2. Take $\bar{x}_{k+1}$ such that $M_{k+1} = f(\bar{x}_{k+1})$. If $\frac{L}{2^{(k+1)\eta}} < \varepsilon$ take $\bar{x} := \bar{x}_{k+1}$. Otherwise set $k := k+1$ and return to Step 1.

3. Analyzing the algorithm

We first need the following

Lemma. *Let $W := \prod_{i=1}^n [b_i, b_i + \rho] \subseteq [0, 1]^n$ be a subcube of S with vertices $x_1, \ldots, x_{2^n}$, and let f be η-Lipschitz-continuous on S with constant L. Then*

$$\max_{x \in W} f(x) - \max_{i=1,\ldots,2^n} f(x_i) \leq L\rho^\eta. \tag{3.1}$$

Proof. Let $f(x_0) := \max_{x \in W} f(x)$. Then $\bar{f}(x) := f(x_0) + A(x_0)(x - x_0)$ takes its maximum in W at one point x_i so that

$$\bar{f}(x_i) := \max_j \bar{f}(x_j) \geq \bar{f}(x_0) = f(x_0)$$

and

$$|f(x_j) - \bar{f}(x_j)| \leq L\|x_0 - x_j\|^\eta < L\rho^l,$$

from which the result follows. □

For every cube W with side length $\frac{1}{2^k}$ and with $f(x_i) < M_k - \frac{L}{2^{k\eta}}$ for all vertices x_i, by the lemma we have $f(x) < M_k \leq y^*$ for all $x \in W$. It follows that $x^* \in I_k$ for all $x^* \in S$ with $f(x^*) = y^*$. Therefore, we obtain $|M_k - y^*| \leq \frac{L}{2^{k\eta}} \leq \varepsilon$ if $k := \lceil \frac{1}{\eta} \log_2 \frac{L}{\varepsilon} \rceil$, where $\lceil x \rceil$ denotes the least integer larger or equal to x.

Hence, one has to run through the algorithm $\lceil \frac{1}{\eta} \log_2 \frac{L}{\varepsilon} \rceil$ times. As a measure for the complexity of the algorithm we have to consider the number of function evaluations needed. Trivially, in any case, at most $(2^k + 1)^n$ evaluations are needed to determine $I_0, I_1, \ldots, I_k$. So in the worst case we need $N = (2^k + 1)^n$ points to achieve an error less than

$$\frac{L}{2^{k\eta}} = \frac{L}{(N^{1/n} - 1)^\eta}, \tag{3.2}$$

which is the usual rate in nonadaptive Quasi-Monte-Carlo-optimization (see [4] or [6]).

In many cases we will now get much better estimates. The quality of these estimates depends on the behavior of f in the neighborhood of its maximum points. From now on we make use of the following three assumptions:

(A) There are at most finitely many points $x_1^*, \ldots, x_w^*$ with $f(x_i^*) = y^*$.

(B) If

$$\tilde{y} := \sup\{f(x) \mid x \text{ is point of a local extremum of } f \text{ and } x \neq x_i^* \text{ for all } i\},$$

then let $b := y^* - \tilde{y} > 0$.

(C) There are constants $\delta, c, \gamma > 0$ with $y^* - f(x) \geq c\|x_i^* - x\|^\gamma$ for all x with $\|x_i^* - x\| < \delta$ and every $i = 1, \dots, w$.

We will show

Theorem. *With the above notation let I_k be a dyadic cube such that for every point $x \in I_k$ the estimate $y^* - f(x) < \varepsilon$ holds. For the determination of I_k by Algorithm 2 we need at most $K(\varepsilon, f)$ function evaluations, where*

$$K(\varepsilon, f) = \begin{cases} \mu_1 + 2w \max\left(\mu_2(\frac{L}{\varepsilon})^{n\frac{\gamma-\eta}{\gamma\eta}}, 3^n \left\lceil \frac{1}{\eta}\log_2 \frac{L}{\varepsilon}\right\rceil\right) & \text{if } \eta < \gamma, \\ \mu_1 + 2w\mu_3 \left\lceil \frac{1}{\eta}\log_2 \frac{L}{\varepsilon}\right\rceil & \text{if } \eta \geq \gamma \end{cases}$$

with

$$\mu_1 = \mu_1(f, n) = \max\left(\left(2\left(\frac{2L}{\min(b, c(\frac{\delta}{2})^\gamma)}\right)^{\frac{1}{\eta}} + 1\right)^n, \left(\frac{n}{\delta} + 1\right)^n\right),$$

$$\mu_2 = \mu_2(f, n) = 2^{n(3-\frac{2\eta}{\gamma})}\left(\frac{2L}{c}\right)^{\frac{n}{\gamma}},$$

$$\mu_3 = \mu_3(f, n) = \max\left(2\left(\frac{2L}{c}\right)^{\frac{n}{\gamma}} 3^n\right).$$

Furthermore, $\{x_1^*, \dots, x_w^*\} = \bigcap_{k=1}^\infty I_k$, and for every $x \in I_k$ there is an $i \in \{1, \dots, w\}$ with $\|x - x_i^*\| < (\frac{2\varepsilon}{c})^{\frac{1}{\gamma}}$ provided that

$$k \geq \max\left(\left\lceil \frac{1}{\eta}\log_2 \min\left(b, c\left(\frac{\delta}{2}\right)^\gamma\right)\right\rceil, \left\lceil \log_2 \frac{2}{\delta}\right\rceil\right).$$

Proof. Let k_0 be such that $\frac{1}{2^{k_0}} \leq \frac{\delta}{2}$ and $M_{k_0} - \frac{L}{2^{k_0\eta}} > \max(\tilde{y}, y^* - c(\frac{\delta}{2})^\gamma)$. Since $M_{k_0} \geq y^* - \frac{L}{2^{k_0\eta}}$, this is satisfied for

$$k_0 := \max\left(\left\lceil \frac{1}{\eta}\log_2 \frac{2L}{\min\left(b, c\left(\frac{\delta}{2}\right)^\gamma\right)}\right\rceil, \left\lceil \log_2 \frac{2}{\delta}\right\rceil\right).$$

Thus, for the determination of $I_0, \dots, I_{k_0}$ at most

$$\max\left(\left(2\left(\frac{2L}{\min\left(b, c\left(\frac{\delta}{2}\right)^\gamma\right)}\right)^{\frac{1}{\eta}} + 1\right)^n, \left(\frac{4}{\delta} + 1\right)^n\right)$$

function evaluations are needed.

Let now $k > k_0$. Then all vertices of subcubes of I_k are in a δ-neighborhood of one of the x_i^*. This follows from (C), since for all x with $\|x_i^* - x\| > \frac{\delta}{2}$ ($i = 1, \dots, w$) we have

$$f(x) < \max\left(\tilde{y}, y^* - c\left(\frac{\delta}{2}\right)^\gamma\right).$$

Note that the subcubes of I_k have sidelength $\frac{1}{2^k} < \frac{\delta}{2}$ and each of these subcubes has at least one vertex x_i with

$$f(x_i) \geq M_k - \frac{L}{2^{k\eta}} \geq M_{k_0} - \frac{L}{2^{k_0\eta}} > \max\left(\tilde{y}, y^* - c\left(\frac{\delta}{2}\right)^\gamma\right).$$

Therefore, for $k > k_0$ we have for at least one vertex x_i of every subcube of I_k

$$y^* - f(x_i) \leq \frac{2L}{2^{k\eta}}.$$

Thus, we obtain

$$\frac{2L}{2^{k\eta}} \geq y^* - f(x_j) \geq c\|x_i^* - x_j\|^\gamma,$$

which yields

$$\|x_i^* - x_j\| \leq \left(\frac{2L}{c2^{k\eta}}\right)^{\frac{1}{\gamma}}$$

for some $i = 1, \dots, w$, and the second assertion of the theorem follows. Furthermore, I_k contains at most

$$w\left(2^{k+1}\left(\frac{2L}{c2^{k\eta}}\right)^{\frac{1}{\gamma}} + 3\right)^n \leq 2n\left(\max\left(2^{k+1}\left(\frac{2L}{c2^{k\eta}}\right)^{\frac{1}{\gamma}}, 3\right)\right)^n$$

vertices of subcubes.

Therefore, for the determination of $I_1, \dots, I_k (k > k_0)$ we need at most

$$\mu_1(f, n) + 2w \sum_{j=k_0+1}^{k} \left(\max\left(2^{j+1}\left(\frac{2L}{c2^{j\eta}}\right)^{\frac{1}{\gamma}}, 3\right)\right)^n$$
$$\leq \begin{cases} \mu_1(f, n) + 2w\max\left(2^n\left(\frac{2L}{c}\right)^{n/\gamma} 2^{n(k+1)(1-\eta/\gamma)}, 3^n k\right) & \text{if } \eta \leq \gamma, \\ \mu_1(f, n) + 2w\max\left(\left(\frac{2L}{c}\right)^{n/\gamma} 2k, 3^n k\right) & \text{if } \eta > \gamma \end{cases}$$

function evaluations. Setting $k := \lceil \frac{1}{\eta} \log_2 \frac{L}{\varepsilon} \rceil$ completes the proof. □

Remark. The following relations between the number N of function evaluations and the error term $y^* - f(\bar{x}) =: \Delta$ are satisfied:

$$\Delta \leq pe^{-N} \quad \text{if } \eta \geq \gamma \tag{3.3}$$

and

$$\Delta \leq q\frac{1}{N^{\frac{\eta\gamma}{n(\gamma-\eta)}}} \quad \text{if } \eta < \gamma \tag{3.4}$$

with constants $p = p(f, n)$ and $q = q(f, n)$. This means that our algorithm is exponentially fast in function evaluations (even in the "worst case" considered in [8]).

4. Concluding remarks and numerical examples

In this final section we add some general remarks concerning the strength and usefulness of our method. First of all, note that in [8] the functions were assumed to be concave, which is much more restrictive than a Lipschitz condition. Clearly, our Lipschitz conditions are global ones. However, it is sufficient to know a Lipschitz constant in a suitable box containing the maximum point (in the case that it is unique).

In the description of Algorithm 2 we restricted ourselves to the simplest case, where the functions are defined on hypercubes. In the following we discuss briefly how to proceed in the case of a function on a compact region $S \subseteq \mathbb{R}^n$. The main idea is to replace the sequence of vertices of axis-parallel subcubes by a fixed "well-distributed" point set $P = \{p_1, \dots, p_m\} \subseteq S$. An appropriate measure for the distribution of points in S is the so-called dispersion

$$D_m(P) = \max_{x \in S} \min_{j=1,\dots,m} d(x, p_j), \tag{4.1}$$

where d denotes a suitable metric on S (e.g., the Euclidean one). For the description of the algorithm in the general case we confine ourselves to the case that there is a unique maximum point x^* (from the presentation of the theorem it is clear how to proceed in the case of finitely many maximum points).

In the first step we have to determine those points in P, which are possible candidates for containing x^* in a $D_m(P)$-neighborhood (a precise description of these points involves the Lipschitz condition and runs along the same lines as the description of Algorithm 2). In a further step we take a homothetic image of P in each of these neighborhoods and repeat the first step. This procedure is iterated until a prescribed stopping condition is satisfied.

The order of convergence of this algorithm substantially depends on the dispersion. For this purpose point sets with low dispersion are needed. Such point sets, in connection with Quasi-Monte-Carlo-optimization, were considered in a series of papers (cf. [3, 4, 6]). In these papers constructions of such sequences and estimates for their dispersion are established. In general,

$$D_m(P) \asymp \frac{1}{m^{\frac{1}{n}}} \tag{4.2}$$

holds. In C. Biester's Ph.D. thesis [1] the above algorithm is studied in the case of various compact regions like balls, cylinders, and special polyhedra. Furthermore, constructions of sequences with low dispersion in those regions are presented. For a detailed survey of the literature on Quasi-Monte-Carlo-Optimization we refer to Chapter 6 of the recent monograph [5] by H. Niederreiter.

In the case of optimization problems with constraints there are two possibilities for applying our algorithm: The first method is to consider the constraints as a compact manifold embedded in some Euclidean space. This manifold can be taken as the compact region S in the above description; as a metric on S one can use the geodesic metric. The case where the manifold is a sphere is extensively studied in [11] and [12]. The second method is to use transformations of this manifold to the unit cube. Such transformation methods were studied in detail by Niederreiter and Peart [7].

In the following we present some numerical examples demonstrating possible applications of Algorithm 2. We note that a MODULA-code of the algorithm is listed in [1] and a PASCAL-code is available by the second author. The first example is a simple "school problem" and shows that, of course, in such cases the algorithm works very well. The subsequent examples seem to be much more interesting. Example 2 is a constrained problem in 3 dimensions which sometimes is used as a test example in global optimization (cf. [2]). Note that our method can be used for constrained problems just by assigning some fixed value to the objective function outside the constraint. Example 3 (Colville No. 4) and Example 4 (Banana-function in 5 dimensions) are also well-known test problems in global optimization (cf. [2, 9]). A reasonable acceleration of the algorithm can be obtained in these cases by a pre-search localizing some tight neighborhood of the extreme value. The other examples are nondifferentiable functions up to dimension 6. The computations show that also for such functions the algorithm works well (which can be considered as one main advantage of Algorithm 2). In the following tables, M denotes the exact value of the maximum, $\tilde{M}$ the approximation by Algorithm 2 after k steps with the help of ℓ generated cubes. The number of function evaluations clearly is $2^n\ell$. We stopped the computations if we reached a good approximation of the maximum or if too many cubes were needed. Finally, we remark that in [1] also a slightly different Algorithm 2* is studied. This algorithm is based on a point sequence of small dispersion and the approximate evaluation of $\operatorname{grad} f$.

Example 1.

$$f(x_1, x_2, x_3) = -(x_1 - 0.567)^2 - (x_2 - 0.89)^2 - (x_3 - 0.123)^2,$$
$$M = 0.00000E + 00$$

k	$\tilde{M}$	ℓ
1	-2.14718E-01	1
5	-2.49249E-04	889
10	-7.45346E-07	5377
15	-6.92149E-10	10201
20	-1.30257E-12	15105
25	-4.99600E-16	19921
26	-5.55112E-17	20825
27	-5.55112E-17	21369
28	0.00000E+00	22009

Example 2 (cf. [2, p. 60]).

$$f(x_1, x_2, x_3) = x_1x_2x_3$$

under the constraints

$$72 - x_1 - 2x_2 - 2x_3 \geq 0, \quad 0 \leq x_i \leq 42, \qquad i = 1, 2, 3,$$
$$M = 3.45600E + 03$$

k	$\tilde{M}$	ℓ
1	0.00000E+00	1
2	0.00000E+00	9
3	2.31525E+03	73
4	2.89406E+03	529
5	3.25582E+03	593
6	3.29652E+03	721
7	3.38809E+03	873
8	3.43514E+03	1113
9	3.43579E+03	1529
10	3.44756E+03	1681
11	3.45346E+03	2097
12	3.45347E+03	2721
13	3.45495E+03	3137
14	3.45568E+03	4257
15	3.45568E+03	7265
16	3.45587E+03	10753
17	3.45596E+03	16705
18	3.45596E+03	28065
19	3.45598E+03	51817
20	3.45600E+03	54217

Example 3 (cf. [2, p. 61]).

$$f(x_1, x_2, x_3, x_4) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2$$
$$+ 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1),$$
$$M = 0.00000E + 00$$

k	$\tilde{M}$	ℓ
1	1.60580E-01	1
2	1.19175E-01	17
3	8.34913E-04	177
4	8.34913E-04	1073
5	8.34913E-04	1329
6	3.48864E-04	1585
7	2.64081E-05	1841
8	2.64081E-05	2705
9	6.55810E-07	2961
10	6.55810E-07	3217
11	4.59164E-07	3601
12	4.27281E-09	4993
13	4.27281E-09	6561
14	4.27281E-09	8225
15	7.41807E-10	14257

Example 4 (5-dimensional Banana-function, cf. [9]).

$$f(x_1, x_2, x_3, x_4, x_5) = \sum_{k=1}^{4} \left(100(x_{k+1} - x_k^2)^2 + (1 - x_k)^2\right) ,$$
$$M = 0.00000E + 00$$

k	$\tilde{M}$	ℓ
1	8.47812E-01	1
2	1.46382E-03	33
3	1.46382E-03	1057
4	1.46382E-03	2081
5	1.46382E-03	3105
6	6.04788E-04	4129
7	4.60453E-05	6209
8	4.60453E-05	9793
9	1.13945E-06	10817
10	1.13945E-06	13729
11	8.03054E-07	14753
12	7.44058E-09	15777
13	7.44058E-09	18881
14	7.44058E-09	19905
15	1.29177E-09	21697
16	6.32966E-10	27329

Example 5.

$$f(x_1, x_2, x_3, x_4) = 0.2620 - \sum_{i=1}^{4} (|x_i - 0.2472| + |x_i - 0.7679| - |x_i - 0.3127|) ,$$
$$M = 0.00000E + 00$$

k	$\tilde{M}$	ℓ
1	-9.28400E-01	1
5	-7.16000E-02	1297
10	-1.28752E-03	3873
15	-6.68168E-05	7233
20	-1.96695E-06	10081
22	-5.96046E-08	11633
23	-5.96046E-08	13025
24	-5.96046E-08	13281
25	-5.96046E-08	13537
26	-5.96046E-08	13793

Example 6.

$$\begin{aligned} f(x_1, x_2, x_3, x_4) = {} & |0.8974 - |x_1 - 0.3172|| - |x_1 + x_2 - 0.5431| \\ & - |x_1 + x_2 + x_3 - 0.7185| - |x_1 + x_2 + x_3 + x_4 - 0.8434|, \\ & M = 0.89740E + 00 \end{aligned}$$

k	$\tilde{M}$	ℓ
1	-3.14800E-01	1
2	2.96400E-01	17
3	6.62200E-01	241
4	7.33400E-01	1249
5	8.11400E-01	2513
6	8.80250E-01	3617
7	8.88325E-01	4401
8	8.89913E-01	4721
9	8.95875E-01	5073
10	8.95875E-01	5457
11	8.96486E-01	5713
12	8.96709E-01	6033
13	8.97114E-01	6385
14	8.97308E-01	6721

Example 7.

$$\begin{aligned} f(x_1, x_2, x_3, x_4) = {} & |1 - |x_1 - 0.3172|| - |1 - |x_1 + x_2 - 0.5431|| \\ & - |x_1 + x_2 + x_3 - 0.7185| - |x_1 + x_2 + x_3 + x_4 - 0.8434|, \\ & M = 2.00000E + 00 \end{aligned}$$

k	$\tilde{M}$	ℓ
1	7.87800E-01	1
2	1.39900E+00	17
3	1.76480E+00	241
4	1.83600E+00	1249
5	1.91400E+00	2513
6	1.98285E+00	3617
7	1.99093E+00	4401
8	1.99251E+00	4721
9	1.99848E+00	5073
10	1.99848E+00	5457
11	1.99909E+00	5713
12	1.99931E+00	6033
13	1.99971E+00	6385
14	1.99991E+00	6721

Example 8.

$$f(x_1, x_2, x_3, x_4, x_5) = 0.3275 - \sum_{i=1}^{5} (|x_i - 0.2472| + |x_i - 0.7679| - |x_i - 0.3127|) ,$$

$$M = 0.00000E + 00$$

k	$\tilde{M}$	ℓ
1	-1.16050E+00	1
2	-1.16050E+00	33
3	-8.95000E-02	1057
4	-8.95000E-02	4641
5	-8.95000E-02	5665
6	-6.67501E-02	6689
7	-1.13750E-02	10273
8	-1.13750E-02	15425
9	-8.15639E-03	16449
10	-1.60941E-03	17473
11	-1.60941E-03	22625
12	-8.32170E-04	23649
13	-3.88712E-04	24673
14	-2.21819E-04	29825
15	-8.35359E-05	33409

Example 9.

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = 0.3852 - \sum_{i=1}^{6} (|x_i - 0.2472| + |x_i - 0.7679| - |x_i - 0.3127|) ,$$

$$M = 0.00000E + 00$$

k	$\tilde{M}$	ℓ
1	-1.47540E+00	1
2	-1.47540E+00	65
3	-2.46001E-02	4161
4	-2.46001E-02	20545
5	-2.46001E-02	24641
6	-2.46001E-02	28737
7	-2.46001E-02	32833

Bibliography

1. C. Biester, *Zahlentheoretische Simulation von Zufallspunkten mit Anwendungen in der numerischen Analysis*, Ph.D. Thesis, Technical University of Vienna, 1991.
2. W. Hock and K. Schittkowski, *Test examples for nonlinear programming codes*, Springer Lecture Notes in Economics and Mathematical Systems, vol. 187, 1981.
3. H. Niederreiter, *A Quasi-Monte-Carlo-Method for the approximate computation of the extreme values of a function*, Studies in Pure Mathematics (P. Erdös, ed.), Birkhäuser, Basel, 1983, pp. 523–529.
4. ——, *Quasi-Monte-Carlo-Methods for global optimization*, Proc. 4th Pannonian Sympos. Math. Stat., Bad Tatzmannsdorf (Austria), 1983, pp. 251–267.
5. ——, *Random number generation and Quasi-Monte-Carlo methods*, SIAM Lecture Notes, vol. 63, Philadelphia, PA, 1992.
6. H. Niederreiter and P. Peart, *Localization of search in Quasi-Monte-Carlo-Methods for global optimization*, SIAM J. Sci. Statist. Comput. **7** (1986), 660–664.
7. ——, *Quasi-Monte-Carlo-Optimization in general domains*, Caribbean J. Math. **4** (1985), 67–85.
8. N. Patel, R. Smith, and Z. Zabinsky, *Pure adaptive search in Monte-Carlo-Optimization*, Math. Programming **43** (1988), 317–328.
9. K. Schittkowski, *More test examples for nonlinear programming codes*, Springer Lecture Notes in Economics and Mathematical Systems, vol. 282, 1987.
10. R. Smith and Z. Zabinsky, *Pure adaptive search in global optimization*, Math. Programming **53** (1992), 323–338.
11. R. F. Tichy, *Random points on the sphere with applications to numerical analysis*, Z. Angew. Math. Mech. **70** (1990), T642–T646.
12. ——, *Random points in the cube and on the sphere with applications to numerical analysis*, J. Comput. Appl. Math. **31** (1990), 191–197.

(C. Biester) Albertg. 49/8, 1080 Wien, Austria

(P.J. Grabner and R.F. Tichy) Institut für Mathematik, TU Graz, Steyrergasse 30, 8010 Graz, Austria
E-mail address, P. J. Grabner: `grabner@ftug.dnet.tu-graz.ac.at`

(G. Larcher) Institut für Mathematik, Universität Salzburg, Hellbrunnerstr. 34, 5020 Salzburg, Austria