A New Quasi-Monte Carlo Algorithm for Numerical Integration of Smooth Functions

Emanouil I. Atanassov, Ivan T. Dimov, and Mariya K. Durchova

Central Laboratory for Parallel Processing, Bulgarian Academy of Sciences
Acad. G. Bontchev, Bl. 25A, 1113 Sofia, Bulgaria
emanouil@parallel.bas.bg, ivdimov@bas.bg, mabs@parallel.bas.bg

Abstract. Bachvalov proved that the optimal order of convergence of a Monte Carlo method for numerical integration of functions with bounded kth order derivatives is $O\left(N^{-\frac{k}{s}-\frac{1}{2}}\right)$, where s is the dimension. We construct a new Monte Carlo algorithm with such rate of convergence, which adapts to the variations of the sub-integral function and gains substantially in accuracy, when a low-discrepancy sequence is used instead of pseudo-random numbers.
Theoretical estimates of the worst-case error of the method are obtained. Experimental results, showing the excellent parallelization properties of the algorithm and its applicability to problems of moderately high dimension, are also presented.

1 Introduction

The paper proposes new quasi-Monte Carlo algorithm for integrating smooth functions. We consider integration over the unit cube $\mathbb{E}^s = [0,1)^s$. The class of functions under consideration is the following:

Definition 1. *For given integers s and k, $s \geq 1$ and $k \geq 0$ the class $W^k(M, \mathbb{E}^s)$ consists of all real functions defined in $\mathbb{E}^s$, such that all the derivatives*

$$\frac{\partial^r f}{\partial x_1^{i_1} \dots \partial x_s^{i_s}}$$

exist for all $i_1 + \dots + i_s = r \leq k$ and there absolute values are bounded by M.

The results of Bachvalov [2, 3] establish lower bounds on the integration error of both deterministic and stochastic or Monte Carlo methods. Various Monte Carlo methods for approximate integration of such functions with the optimal order of convergence $\mathrm{O}\left(N^{-\frac{1}{2}-\frac{k}{s}}\right)$ are known (see, e.g. [1, 5, 10, 20]). In this paper we investigate from theoretical and practical view point a quasi-Monte Carlo algorithm, based loosely on the same ideas. The idea of quasi-Monte Carlo algorithms is to replace the pseudo-random numbers, used in Monte Carlo, with a deterministic sequence of points, uniformly distributed in some high-dimensional unit cube $\mathbb{E}^s$. The quality of distribution of these sequences is measured through

I. Lirkov et al. (Eds.): LSSC 2003, LNCS 2907, pp. 128–135, 2004.

their discrepancy. Infinite sequences with order of the discrepancy $\mathrm{O}\left(\frac{\log^s N}{N}\right)$, which is believed to be the optimal, are called low-discrepancy sequences. Note that in Monte Carlo methods one can frequently obtain a statistical error estimate along with the result, which is necessary in many areas. In quasi-Monte Carlo methods a similar error estimate of statistical nature can be obtained using the so-called scrambling of sequences, which is a way of adding some randomness to otherwise entirely deterministic sequences. The most popular quasi-Monte Carlo integration method is based on the simple formula:

$$\int\limits_{\mathbb{E}^s} f\,(x)\;dx \approx \frac{1}{N}\sum_{j=1}^{N} f(x_j),$$

where $\sigma = \{x_j\}_{j=1}^{\infty}$ is a uniformly distributed sequence. For a good introduction in the theory of uniform distribution modulo 1 see, e.g., the books [9] and [7]. More complex methods are developed and tested by many authors (see, e.g., [18, 16]). A method, based on scrambled nets, can be seen in [12]. In quasi-Monte Carlo methods it is important to achieve low constructive dimensionality (for a definition, see [17], p. 255). Since the first few coordinates of a low-discrepancy sequence are better distributed, various ways to adjust the integration procedure to the properties of the integrand are proposed (see, e.g., the Brownian bridge construction in [4]).

The performance of the quasi-Monte Carlo method depends on the quality of the distribution of the underlying low-discrepancy sequence. Various families of such sequences are known, and we tested some of the most popular ones in our algorithm. The algorithm is described in Sec. 2. Numerical results, showing the applicability of the method and its excellent parallelization properties, are given in Sec. 3.

2 Description of the Algorithm

Since we are going to describe our algorithm using the term *pseudo-inverse matrix*, we provide the following

Definition 2. *(see e.g. [11], p. 257) Let A be an $m \times n$ real matrix with rank r. Suppose that $U^T A V = \Sigma$ is the SVD of A. Then the pseudo inverse matrix of A is defined as $A^+ = V\Sigma + U^T$, where*

$$\Sigma^+ = diag\left(\frac{1}{\sigma_1}\ldots,\frac{1}{\sigma_r},0,\ldots,0\right) \in R^{n\times m}$$

is referred to as the pseudo inverse of A. If $rank(A) = n$, then $A^+ = (A^T A)^{-1}A^T$.

Definition 3. *Let t be an integer, $t \geq 1$, and let $a_1, \ldots, a_t$ be fixed points in $\mathbb{E}^s$. Let $f \in W^k(M, \mathbb{E}^s)$ for some M. The s-tuples $(i_1, \ldots, i_s)$ with $i_1 + \cdots + i_s = r < k$*

can be ordered lexicographically and there are exactly $\binom{s+k-1}{k-1}$ *of them. Consider the matrix* B *with* t *rows and* $\binom{s+k-1}{k-1}$ *columns, such that the* $n^{-\text{th}}$ *column of* B *corresponds to the* $n^{-\text{th}}$ *tuple* $(i_1, \ldots, i_s)$ *and contains all the products*

$$b_j\,(i_1, \ldots, i_s) = \left(a_j^{(1)}\right)^{i_1} \ldots \left(a_j^{(s)}\right)^{i_s}, \qquad for \quad j = 1, \ldots, t.$$

Suppose that $t \geq \binom{s+k-1}{k-1}$ *and that* B *has rank* $\binom{s+k-1}{k-1}$*. Let the matrix* C *be the pseudo-inverse of* B*.* C *has* $\binom{s+k-1}{k-1}$ *rows and* t *columns. Now the interpolation polynomial* $L\,(f, x)$ *is defined by*

$$L\,(f, x) = \sum_{j=1}^{t} f(a_j) \sum_{(i_1,\ldots,i_s), i_1+\cdots+i_s<k} c_j\,(i_1, \ldots, i_s)\, x_1^{i_1} \ldots x_s^{i_s}$$

Observe that if the points $a_1, \ldots, a_s$ are in general position, the matrix B has rank $\binom{s+k-1}{k-1}$. Since in our definition B has always full rank, the pseudo-inverse C is equal to $(B(B^TB)^{-1})^T$. The reason for using formulae with $t > \binom{s+k-1}{k-1}$ for numerical integration is that the coefficients of the matrix C are much smaller in this case. Of all the matrices satisfying $BAB = B$ the pseudo-inverse is chosen because its Frobenius norm $\| \, . \, \|_F$ is the smallest.

Now let us fix some integers k and s, greater than 1 and consider functions $f \in W^k\,(M, \mathbb{E}^s)$ defined over the unit cube $\mathbb{E}^s$. We also fix the number t and the points $\{a_1, \ldots, a_t\} \subset \mathbb{E}^s$, such that an interpolation formula $L(f)(x) = L(f, x)$ as described in Definition 3 is defined. Observe that by integrating over $\mathbb{E}^s$ we get a quadrature formula

$$\int\limits_{\mathbb{E}^s} f\,(x)\,dx \approx \sum_{j=1}^{t} r_j f(a_j),$$

where

$$r_j = \sum_{(i_1,\ldots,i_s), i_1+\cdots+i_s<k} \frac{c_j\,(i_1, \ldots, i_s)}{(i_1+1)\ldots(i_s+1)}.$$

Definition 4. *Consider a rectangular area* $K \subset \mathbb{E}^s$ *and an interpolation formula* $L : f \to L(f)$ *as in definition 3. Let* T *be a linear transformation such that* $T(K) = \mathbb{E}^s$*. For a given function* $f : K \to R$ *consider the function* $g : \mathbb{E}^s \to R$ *such that* $g(y) = f(Tx)$*. The interpolation formula* $L_K : f \to L_K(f)$ *is defined by* $L_K(f, x) = L(g, Tx)$*.*

Now we define a Monte Carlo integration formula, which we are going to investigate in the sequel.

Definition 5. *Let* $N \geq 1$ *be an integer. Consider a representation of the unit cube* $\mathbb{E}^s$ *as a union of* N *rectangles* $K_1, \ldots, K_N$*.*

For the rectangle $K_i = \prod_{j=1}^{s} [b_j, c_j)$ *we consider the simplest linear transformation* T *such that* $T(K_i) = \mathbb{E}^s$*:*

$$x_j \to \frac{x - b_j}{c_j - b_j}.$$

By T^{-1} *the points* $\{a_1, \ldots, a_t\}$ *are transformed into points* $\left\{a_1^{(i)}, \ldots, a_t^{(i)}\right\} \subset K_i$*. Let* m *be a given integer,* $m \geq 2$*, and let* $\xi^{(1)}, \ldots, \xi^{(N)}$ *be* N *independent random variables, uniformly distributed in the respective rectangles* $K_1, \ldots, K_N$*. For every* K_i *we use* m *samples* $\xi_1^{(i)} \ldots \xi_m^{(i)}$ *of* $\xi^{(i)}$ *and define the Monte Carlo estimate of* $\int_{K_i} f(x)\, dx$ *to be*

$$vol\,(K_i) \left(\sum_{j=1}^{t} r_j f(a_j^{(i)}) + \frac{1}{m} \sum_{j=1}^{m} f\left(\xi_j^{(i)}\right) - L_K\left(f, \xi_j^{(i)}\right) \right).$$

The Monte Carlo estimate for the integral $\int_{\mathbb{E}^s} f(x)\, dx$ *is obtained by summing all the estimates for the integrals* $\int_{K_i} f(x)\, dx$*.*

Such integration formulae were considered in [1], but only when t has the smallest possible value $\binom{s+k-1}{k-1}$. This additional degree of freedom results in faster convergence of the resulting formulae.

3 Numerical Results

In this section we present numerical results that are obtained for calculating the value of the integral

$$I_k = \int_{\mathbb{E}^s} F_k(x) dx, \quad k = 1, \ldots, 7.$$

The functions, that are considered here, are with different peculiarities. Often they are used for benchmarking of Monte Carlo and quasi-Monte Carlo algorithms. They are given as follows:

$$F_1 = \prod_{i=1}^{s} \left(x_i^3 + \frac{3}{4} \right),$$

$$F_2 = \sum_{i=1}^{s} \prod_{j=1}^{i} (-1)^j x_j,$$

$$F_3 = \exp(-x^2) \cos(|x|),$$

$$F_4 = \exp(-x^2)\sqrt{(1+x^2)},$$

$$F_5 = \cos\left(2\pi u_1 + \sum_{i=1}^{s} a_i x_i\right),$$

$$F_6 = \left(1 + \sum_{i=1}^{s} a_i x_i\right)^{-(s+1)},$$

$$F_7 = \exp\left(-\sum_{i=1}^{s} a_i^2 (x_i - u_i)^2\right).$$

The first function F_1 has been used in [8] and [14] as a test in a context of rating point sets for quasi-Monte Carlo and Monte Carlo integration. The last three functions are a part of the test functions package proposed by Genz or in [13, 15, 19]. The parameters u_1, u_i, a_i are divided into unaffective and affective parameters. The vector $\boldsymbol{a}$ of affective parameters is scaled so that it satisfies the requirement

$$\|\boldsymbol{a}\|_1 = 110 s^{-\frac{3}{2}},$$

$$\|\boldsymbol{a}\|_1 = 600 s^{-2},$$

$$\|\boldsymbol{a}\|_1 = 100 s^{-1}$$

for the functions F_5, F_6, F_7 respectively, that depends only on the space dimensionality.

In Table 1 we present numerical results from the approximate integration of the integral I_1 using the Monte Carlo and quasi-Monte Carlo version of the algorithm. The algorithms where tested for dimension 4, 5, 6, and 9. We did not compare our algorithm with crude Monte Carlo as in [6], because of its much slower convergence. From this table we made the conclusion, that the quasi-Monte Carlo variant of the algorithm shows approximately the same accuracy as the Monte Carlo algorithm, i.e., $O(N^{-\frac{1}{2}-\frac{k}{s}})$, for smooth functions.

Table 1. Results for I_1, with dimension s, number of steps N and number of points per cube 40.

N	r	$s=4$		$s=5$		$s=6$		$s=9$	
		MC	S.	MC	S.	MC	S.	MC	S.
3	4	2.69E-5	1.52E-5	2.65E-5	1.71E-5	2.43E-5	2.53E-5	1.34E-5	9.46E-6
	6	4.59E-7	2.61E-7	1.05E-6	7.07E-7	2.62E-7	2.31E-7	1.57E-6	4.83E-7
4	4	5.59E-6	2.84E-6	2.68E-7	2.78E-6	2.33E-6	2.65E-6	1.36E-6	1.74E-6
	6	7.06E-8	3.07E-8	4.99E-8	7.15E-8	5.84E-8	4.11E-8	-	-
5	4	1.43E-6	1.15E-6	7.51E-7	5.41E-7	5.15E-7	5.67E-7	-	-
	6	5.30E-9	3.57E-9	1.47E-8	1.08E-8	8.19E-9	8.39E-9	-	-

Table 2. Numerical results for I_2, I_3, I_4 with dimension $s = 5$.

N	r	I_2		I_3		I_4	
		MC	S.	MC	S.	MC	S.
3	4	1.07E-6	1.86E-6	1.29E-6	2.50E-6	7.93E-7	1.74E-6
	6	4.07E-16	8.15E-16	2.52E-8	3.37E-8	1.89E-8	2.45E-8
4	4	1.86E-7	3.05E-7	4.28E-7	1.86E-7	2.91E-7	1.19E-7
	6	4.12E-16	4.28E-16	2.55E-9	3.52E-9	1.91E-9	2.51E-9
5	4	3.54E-8	5.70E-8	9.47E-8	6.62E-8	6.92E-8	4.85E-8
	6	5.19E-16	5.19E-16	4.96E-10	7.74E-10	3.72E-10	5.75E-10

Table 3. Numerical results for I_5, I_6, I_7 with dimension $s = 5$.

N	r	I_5		I_6		I_7	
		MC	S.	MC	S.	MC	S.
3	4	6.44E-3	5.89E-3	3.06E-8	4.61E-8	1.93E-5	4.37E-5
	6	9.65E-3	8.12E-3	1.02E-7	3.56E-7	6.24E-5	7.02E-5
4	4	5.05E-3	5.45E-3	3.01E-8	3.05E-8	5.33E-6	8.57E-6
	6	4.51E-3	2.74E-3	6.04E-8	1.01E-7	1.76E-5	1.41E-5
5	4	1.84E-3	3.89E-3	7.68E-9	1.09E-8	5.12E-6	4.25E-6
	6	3.09E-3	2.06E-3	2.80E-8	1.51E-8	4.26E-6	3.96E-6

Table 4. CPU time and efficiency for F_5, dimension $s = 5$, smoothness 4, number of steps 20, number of points per cube 40.

NP	1		2		4		8		16		32	
Method	time	Eff.	Time	Eff.	Time	Eff.	Time	Eff.	Time	Eff.	Time	Eff.
MC	336	1	169	0.99	83	1.01	43	0.98	24	0.88	14	0.75
Sobol	337	1	170	0.99	86	0.98	45	0.92	24	0.88	15	0.71

In Table 2 and 3 we show numerical results for the other integrals in 5 dimensions. We again observe roughly the same probable error of the Monte Carlo and the quasi-Monte Carlo algorithm. All these results are obtained with the parameter m (the number of quasi-random points in every small cube) equal to 40. In Table 4 we show the parallel efficiency of the algorithm. The computations for every small cube can be done in parallel, and thus the algorithm possesses a significant amount of natural parallelism. The results are achieved on a cluster of Intel Xeon 2.2 processors, using MPI.

4 Conclusions

We have developed a quasi-Monte Carlo method, which achieves order of convergence $\mathrm{O}\left(N^{-\frac{1}{2}-\frac{k}{s}}\right)$ on smooth functions. The proposed quasi-Monte Carlo

method shows roughly the same accuracy as the Monte Carlo version, and also exhibits good parallel efficiency. However, the statistical aposteriory error estimation, based on the use of scrambled low-discrepancy sequences, is slightly more complicated and less reliable, than Monte Carlo methods. In our experiments we obtained good results by using the Sobol sequences. Note that the constructive dimensionality of the algorithm is relatively high - sm, and thus the integration method can be used for assessing the quality of distribution of other families of low-discrepancy sequences, i.e., for benchmarking purposes.

Acknowledgements

Supported by the project of European Commission - BIS 21 under contract ICA1-CT-2000-70016 and by the Ministry of Education and Science of Bulgaria under contract NSF I-1201/02 and NSF MM-902/99.

References

1. E. Atanassov, I. Dimov. A new optimal Monte Carlo method for calculating integrals of smooth functions. *Monte Carlo Methods and Applications*, **5** 2 (1999), 149–167.
2. N. S. Bachvalov. On the approximate computation of multiple integrals. *Vestnik Moscow State University, Ser. Mat., Mech.*, 4 (1959) 3–18.
3. N. S. Bachvalov. Average Estimation of the Remainder Term of Quadrature Formulas. *USSR Comput. Math. and Math. Phys.*, **1** 1 (1961) 64–77.
4. R. E Caflisch, W. Morokoff, A. B. Owen. Valuations of mortgage backed securities using Brownian bridges to reduce effective dimension. *Journal of Computational Finance*, **1** (1997) 27–46.
5. S. Capstick, B. D. Keister. *Multi dimensional Quadrature Algorithms at Higher Degree and/or Dimension*, *www.scri.fsu.edu/˜capstick/papers/quad.ps.*
6. I. Dimov, A. Karaivanova, R. Georgieva, S. Ivanovska. Parallel Importance Separation and Adaptive Monte Carlo Algorithms for Multiple Integrals, In: I. Dimov, I. Lirkov, S. Margenov, Z. Zlatev Eds., *Numerical Methods and Applications, Lecture Notes in Computer Science*, **2542**, Springer Verlag (2003) 99–107.
7. M. Drmota, R. F. Tichy. Sequences, Discrepancies and Applications, *Lecture Notes on Mathematics, Springer, Berlin*, 1997, N 1651.
8. K. Entacher, A. Uhl, S. Wegenkittl. Linear congruential generators for parallel Monte Carlo: the leap-frog case, *Monte Carlo Meth. Appl.*, **4** (1998) 1–16.
9. L. Kuipers, H. Niederreiter. *Uniform distribution of sequences*, John Wiley & sons, New York, 1974.
10. N. Kjurkchiev, Bl. Sendov, A. Andreev. Numerical Solution of Polynomial Equations, P.G. Ciarlet and J. L. Lions (Eds.), *Handbook of Numerical Analysis, Solution of Equations in R^n (Part 2)*, **3**, North-Holland, Amsterdam, NY, 1994.
11. C. Van Loan, G. Golub. *Matrix Computations*, The John Hopkins University Press, Baltimore and London, third edition, 1996.
12. A. B. Owen. Scrambled Net Variance for Integrals of Smooth Functions, *Annals of Statistics*, **25** 4 (1997) 1541–1562.
13. A. B. Owen. The dimension distribution and quadrature test functions, *www-stat.stanford.edu/˜owen/reports*, November, 2001.

14. W. Ch. Schmid, A. Uhl. Techniques for parallel quasi-Monte Carlo integration with digital sequences and associated problems, *Math. and Comp. in Sim.*, **55** (2001) 249–257.
15. R. Schuerer. A comparison between (quasi-)Monte Carlo and cubature rule based method for solving high-dimensional integration problems, *Math. and Comp. in Sim.*, **62** 3–6 (2003) 509–517.
16. I. H. Sloan, H. Wozniakowski. When are quasi-Monte Carlo Algorithms efficient in high-dimensional integration, *Journal of Complexity*, 14 (1998) 1–33.
17. I. M. Sobol′. *Monte Carlo Methods*, Nauka, Moscow, 1973.
18. I. M. Sobol′, D. I. Asotsky. One more experiment on estimating high-dimensional integrals by quasi-Monte Carlo methods, *Math. and Comp. in Sim.*, **62** 3–6 (2003) 255–275.
19. *http://www.math.wsu.edu/math/faculty/genz/homepage*, Software/MULTST.
20. M. D. Takev. On Probable Error of the Monte Carlo Method for Numerical Integration, *Mathematica Balkanica (New Series)*, **6** (1992) 231–235.