A new optimal Monte Carlo method for calculating integrals of smooth functions *

Emanouil I. Atanassov [1], Ivan T. Dimov [1],

Abstract

An optimal Monte Carlo method for numerical integration of multi-dimensional integrals is proposed and studied. It is know that the best possible order of the mean square error of a Monte Carlo integration method over the class of the k times differentiable functions of d variables is $O\left(N^{-\frac{1}{2}-\frac{k}{d}}\right)$. In this paper we present two algorithms implementing the method under consideration.

Estimates for the computational complexity are obtained. Numerical tests showing the efficiency of the algorithms are also given.

Keywords: Monte Carlo method, optimal quadrature formula, rate of convergence.

ASM subject classifications: 65C05, 65U05, 65F10, 65Y20.

1 Introduction

In this paper a Monte Carlo method for calculating multidimensional integrals of smooth functions is considered. Let d and k be integers, and $d, k \geq 1$. We consider the class $\mathbf{C}_k^d$ of real functions f defined over the unit cube $\mathbf{E}^d = [0,1)^d$, possessing all the partial derivatives

$$\frac{\partial^r f(x)}{\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}}, \;\; \alpha_1 + \ldots + \alpha_d = r \leq k,$$

which are continuous when $r < k$ and bounded in sup norm when $r = k$. The semi-norm $\|\cdot\|$ on $\mathbf{C}_k^d$ is defined as

$$\|f\| = \sup\left\{\left|\frac{\partial^r f(x)}{\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}}\right| \;\; |\alpha_1 + \ldots + \alpha_d = k, \;\; x \equiv (x_1, ..., x_d) \in \mathbf{E}^d\right\}.$$

*Supported by the Ministry of Science and Education of Bulgaria under Grant # I-501/95.

[1]Central Laboratory for Parallel Processing, Department of Parallel Algorithms, Bulgarian Academy of Sciences, Acad. G. Bonchev St.,bl. 25 A, 1113 Sofia, Bulgaria, e-mail:emanouil@copern.acad.bg, dimov@amigo.acad.bg, Web site: http: //www.acad.bg/BulRTD/math/dimov2.html

There are two classes of methods for numerical integration of such functions over $\mathbf{E}^d-$ *deterministic* and *stochastic* or *Monte Carlo methods*. We consider the following quadrature formula

$$I(f) = \sum_{i=1}^{N} c_i f(x^{(i)}), \tag{1}$$

where $x^{(i)} \equiv \left(x_1^{(i)}, \dots, x_d^{(i)}\right) \in \mathbf{E}^d,\ i = 1, \dots, N$ are the nodes and $c_i,\ i = 1, \dots, N$ are the weights. If $x^{(i)}$ and c_i are real values, the formula (1) defines a deterministic quadrature formula. If $x^{(i)},\ i = 1, \dots, N$ are random points defined in $\mathbf{E}^d$ and c_i are random variables defined in $\mathbf{R}$ the formula (1) defines a Monte Carlo quadrature formula.

The following results of Bahvalov establish lower bounds for the integration error in both cases:

Theorem 1.1 *(Bahvalov [Ba59, Ba61]) There exists a constant $c(d, k)$ such that for every quadrature formula $I(f)$ which is fully deterministic and uses the function values at N points there exists a function $f \in \mathbf{C}_k^d$ such that*

$$\left| \int_{\mathbf{E}^d} f(x)dx - I(f) \right| \geq c(d, k) \, \|f\| \, N^{-\frac{k}{d}}.$$

Theorem 1.2 *(Bahvalov [Ba59, Ba61]) There exists a constant $c(d, k)$ such that for every quadrature formula $I(f)$ which involves random variables and uses the function values at N points there exists a function $f \in \mathbf{C}_k^d$ such that*

$$\left\{ \mathrm{E} \left[\int_{\mathbf{E}^d} f(x)dx - I(f) \right]^2 \right\}^{1/2} \geq c(d, k) \, \|f\| \, N^{-\frac{1}{2}-\frac{k}{d}}.$$

When d is sufficiently large it is evident that methods involving the use of random variables will outperform the deterministic methods.

Monte Carlo methods, which achieve the order $O\left(N^{-\frac{1}{2}-\frac{k}{d}}\right)$ are called *optimal*. In fact methods of this kind are superconvergent (following the definition given by Sobol [So73]) with an unimprovable rate of convergence. It is not an easy task to construct a unified method with such rate of convergence for any dimension d and any value of k. Various methods for Monte Carlo integration that achieve the order $O\left(N^{-\frac{1}{2}-\frac{k}{d}}\right)$ are known. While in the case of $k = 1$ and $k = 2$ these methods are fairly simple and are widely used (see, for example, [So89, SAK94, Ta92, TN74, NR96]), when $k \geq 3$ such methods become much more sophisticated. The first optimal stochastic method was proposed by Mrs. Dupach [Du56] for $k = 1$. This method uses the idea of separation of the domain of integration into uniformly small (according both to the probability and to the sizes) disjoint subdomains and generating one or small number of points in every subdomain. This idea was largely used [SAK94, Ta92] for creation Monte Carlo methods with high rate of convergence.

There exist also the so-called *adaptive* Monte Carlo methods proposed by Lautrup (see, pp. 391-392 in the book of Davis and Rabinowitz [DR84]), which use a priori and/or a posteriori information obtained during calculations. The main idea of this approach is to adapt the Monte Carlo quadrature formula to the element of integration. In [KD97] the idea of separation of the domain into *uniformly small* subdomains is combined with the idea of adaptivity to obtain an optimal Monte Carlo quadrature for the case $k = 1$.

In this work we also combine both ideas - separation and adaptivity and present an optimal Monte Carlo quadrature for any k. We separate the domain of integration into disjoint subdomains. Since we consider the cube $\mathbf{E}^d$ we divide it into $N = n^d$ disjoint cubes $K_j, j = 1, \ldots, N$. In every cube K_j we calculate the coordinates of $\begin{pmatrix} d+k-1 \\ d \end{pmatrix}$ points $y^{(r)}$. We select m uniformly distributed and mutually independent random points from each cube K_j and consider the Lagrange interpolation polynomial of the function f at the point z, which uses the information from the function values at the points $y^{(r)}$. After that we approximate the integral in the cube K_j using the values of the function and the Lagrange polynomial at the m selected random points. Then we sum these estimates over all cubes $K_j,\ j = 1, \ldots, N$. The adaptivity is used when we consider the Lagrange interpolation polynomial. The estimates for the probable error and for the mean square error are proven. It is shown that the presented method has the best possible rate of convergence, i.e. it is an optimal method.

In this work we present two algorithms for Monte Carlo integration that achieve such order of the integration error, along with estimates of their *computational complexity*. It is known that the *computational complexity* is defined as number of operations needed to perform the algorithm on the sequential (von Neumann) model of computer architecture. It is important to be able to compare different Monte Carlo algorithms for solving the same problem with the same accuracy (with the same probable or mean square error). It is possible to show that the computational complexity could be estimated as a product of $tD\xi$, where t is the time needed to calculate one value of the random variable ξ, whose mathematical expectation is equal to the exact value of the integral under consideration and $D\xi$ is the variance (it is shown by I.M. Sobol in [So73]). In our consideration we do not use this presentation and estimate the computational complexity directly as number of floating point operations (flops) used to calculate the approximate value of the integral.

One can also use some other estimators of the quality of the algorithm (if parallel machines are available) like *speed-up* and *parallel efficiency* [DT93]. It is easy to see that the speed-up of our algorithms is linear and the parallel efficiency is close to 1 due to the relatively small communication costs. The numerical tests performed on 2-processor and 4-processor machines confirm this.

The paper is organized as follows. Section 2 contains the description of the method. Here the estimates for the probable error and for mean square error are obtained, showing that the method is optimal. In Section 3 two algorithms implementing the described Monte Carlo method are presented. Estimates of the computational complexity are proven. Section 4 presents some numerical tests confirming the high computational efficiency of the algorithms. The paper ends with some concluding remarks about applicability of the algorithms under consideration.

2 Description of the method and theoretical estimates

Definition 2.1 *Given a Monte Carlo integration formula for the functions in the space* $\mathbf{C}_k^d$ *by* $err(f, I)$ *we denote the integration error*

$$\int_{\mathbf{E}^d} f(x)dx - I(f),$$

by $\varepsilon(f)$ *the probable error meaning that* $\varepsilon(f)$ *is the least possible real number with*

$$P\left(|err(f, I)| < \varepsilon(f)\right) \geq \frac{1}{2}$$

and by $r(f)$ *the mean square error*

$$r(f) = \left\{E\left[\int_{\mathbf{E}^d} f(x)dx - I(f)\right]^2\right\}^{1/2}.$$

For each integer $n, d, k \geq 1$ we define a Monte Carlo integration formula, depending on an integer parameter $m \geq 1$ and $\left(\begin{array}{c} d+k-1 \\ d \end{array}\right)$ points in $\mathbf{E}^d$ in the following way:

The $\left(\begin{array}{c} d+k-1 \\ d \end{array}\right)$ points $x^{(r)}$ have to fulfill the condition that if for some polynomial $P(x)$ of combined degree less than k

$$P\left(x^{(r)}\right) = 0,$$

then $P \equiv 0$. Let $N = n^d$, $n \geq 1$. We divide the unit cube $\mathbf{E}^d$ into n^d disjoint cubes

$$\mathbf{E}^d = \bigcup_{j=1}^{n^d} K_j, \quad \text{where} \quad K_j = \prod_{i=1}^{d} [a_i^j, b_i^j),$$

with $b_i^j - a_i^j = \frac{1}{n}$ for all $i = 1, \ldots, d$. Now in every cube K_j we calculate the coordinates of $\left(\begin{array}{c} d+k-1 \\ d \end{array}\right)$ points $y^{(r)}$, defined by

$$y_i^{(r)} = a_i^r + \frac{1}{n} x_i^{(r)}.$$

Suppose, we select m random points $\xi(j,s) = (\xi_1(j,s),\ldots,\xi_d(j,s))$ from each cube K_j, such that all $\xi_i(j,s)$ are uniformly distributed and mutually independent, calculate all $f(y^{(r)})$ and $f(\xi(j,s))$ and consider the Lagrange interpolation polynomial of the function f at the point z, which uses the information from the function values at the points $y^{(r)}$. We call it $L_k(f,z)$.

For all polynomials P of degree at most $k-1$ we have $L_k(p,z) \equiv z$.

We approximate

$$\int_{K_j} f(x)dx \approx \frac{1}{mn^d}\sum_{s=1}^{m}\left[f(\xi(j,s)) - L_k(f,\xi(j,s))\right] + \int_{K_j} L_k(f,x)dx.$$

Then we sum these estimates over all $j = 1,\ldots,N$ to achieve

$$I(f) \approx \frac{1}{mn^d}\sum_{j=1}^{N}\sum_{s=1}^{m}\left[f(\xi(j,s)) - L_k(f,\xi(j,s))\right] + \sum_{j=1}^{N}\int_{K_j} L_k(f,x)dx.$$

We prove the following

Theorem 2.1 *The quadrature formula constructed above satisfies*

$$\varepsilon(f,k,d,m) \le c'_{d,k}\frac{1}{m}\,\|f\|\,N^{-\frac{1}{2}-\frac{k}{d}}$$

and

$$r(f,k,d,m) \le c''_{d,k}\frac{1}{m}\,\|f\|\,N^{-\frac{1}{2}-\frac{k}{d}},$$

where the constants $c'_{d,k}$ and $c''_{d,k}$ depend implicitly on the points $x^{(r)}$, but not on N.

P r o o f. One can see that

$$E\left\{\frac{1}{mn^d}\sum_{s=1}^{m}\left[f(\xi(j,s)) - L_k(f,\xi(j,s))\right] + \int_{K_j} L_k(f,x)dx\right\} = \int_{K_j} f(x)dx$$

and

$$\begin{aligned} & D\left\{\frac{1}{mn^d}\sum_{s=1}^{m}\left[f(\xi(j,s)) - L_k(f,\xi(j,s))\right] + \int_{K_j} L_k(f,x)dx\right\} \\ = \; & \frac{1}{m}D\left\{\frac{1}{n^d}\left[f(\xi(j,1)) - L_k(f,\xi(j,1))\right]\right\}. \end{aligned}$$

Note that

$$\int_{K_j} L_k(f,x)dx = \frac{1}{n^d}\sum_{0\le i_1+\ldots+i_d\le k-1} A(r)f(y^{(r)}),$$

where the coefficients $A(r)$ are the same for all cubes K_j and depend only on $\left\{x^{(r)}\right\}$. Using Taylor series expansion of f over the center of the cube K_j, one can see that

$$|f(\xi(s,t) - L_k(f,\xi(j,s)| \leq c_{d,k} n^{-k} \|f\|,$$

and therefore

$$D\left\{\frac{1}{n^d}\left[f(\xi(j,s)) - L_k(f,\xi(j,s))\right]\right\} \leq c'_{d,k} n^{-2d} n^{-2k} \|f\|^2$$

Taking into account that the $\xi(j,s)$ are independent, we obtain that

$$D\left\{\sum_{j=1}^{n^d} \frac{1}{mn^d} \sum_{t=1}^{m} \left[f(\xi(j,s)) - L_k(f,\xi(j,s))\right] + \int_{K_j} L_k(f,x)dx\right\} \leq$$

$$n^d \frac{1}{m^2} m c'_{d,k} n^{-2k} n^{-2d} \|f\|^2 = \frac{1}{m} c'_{d,k} n^{-d} n^{-2k} \|f\|^2$$

and therefore ($N = n^d$)

$$r(f,k,d,m) \geq \frac{1}{\sqrt{m}} c(d,k) N^{-\frac{1}{2}-\frac{k}{d}} \|f\|.$$

The application of the Chebishev's inequality yields

$$\varepsilon(f,k,d,m) \leq \frac{1}{\sqrt{m}} c'_{d,k} \|f\| N^{-\frac{1}{2}-\frac{k}{d}}$$

for the probable error ε, where $c'(d,k) = \sqrt{2}c(d,k)$, which concludes the proof $\diamondsuit$

One can replace the Lagrange approximation polynomial with other approximation schemes that use the function values at some fixed points, provided they are exact for all polynomials of degree less than k.

3 Estimates of the computational complexity

In this section two algorithms implementing our method are given. Estimates of the computational complexity of both algorithms are presented.

Algorithm 1 (A.1)

In the fist algorithm the points $x^{(r)}$ are selected so that they fulfill certain conditions that would assure good Lagrange approximation of any function from $\mathbf{C}_k^d$. Let us order all the monomials of d variables and degree less than $k - \mu_1, \ldots, \mu_t$. Note that there are exactly $\left(\begin{array}{c} d+k-1 \\ d \end{array}\right)$ of them. We use a pseudo-random number generator to obtain many sets of points $x^{(r)}$, then we select the one for which the norm of the inverse of the matrix $A = (a_{ij})$ with

$$a_{ij} = \mu_i(x^{(j)})$$

is the smallest one.

Once it is selected for fixed k and d, the same set of points will be used for integrating every functions from the space $\mathbf{C}_k^d$. We do not need to store the co-ordinates of the points $x^{(j)}$, if we can record the state of the generator just before it produces the *"best"* set of $x^{(j)}$. Since the calculations of the elements of the matrix A and its inverse, as well as the coefficients of the interpolation type quadrature formula are made only once if we know the state of the pseudo- random number generator that will produce the set of points $x^{(j)}$, they would counts as $O(1)$ in our estimates for the number of flops used to calculate the integral of a certain function from $\mathbf{C}_k^d$. These calculations could be considered as *preprocessing*. We prove the following

Theorem 3.1 *The computational complexity of the numerical integration of a function from $\mathbf{C}_k^d$ using Algorithm A.1 is estimated by:*

$$N_{fp} \leq N\left[m + \begin{pmatrix} d+k-1 \\ d \end{pmatrix}\right] a_f + mN\left[d\left(b_r + 2\right) + 1\right] + \tag{2}$$

$$N\begin{pmatrix} d+k-1 \\ d \end{pmatrix}\left[2m + 1 + 2d + 2\begin{pmatrix} d+k-1 \\ d \end{pmatrix} + 1\right] + c(d,k) \tag{3}$$

where b_r denotes the number of flops used to produce a uniformly distributed random number in $[0,1)$, a_f stands for the number of flops needed for each calculation of a function value, and $c(d,k)$ depends only on d and k.

P r o o f. As it was pointed out above, the calculation of the coordinates of the points $x^{(r)}$, the elements of A and A^{-1}, and the coefficients of the interpolation type quadrature formula can be considered to be done with $c_1(d,k)$ flops, if we know the initial state of the pseudo-random number generator that produces the set of points $x^{(r)}$. Then in

$$2dN\begin{pmatrix} d+k-1 \\ d \end{pmatrix}$$

operations we calculate the coordinates of the points $y^{(r)}$ in each cube and in

$$mdN(b_r+2)$$

flops we obtain the uniformly distributed random points we shall use. The calculation of the values of f at all these points takes

$$N\left[m+\binom{d+k-1}{d}\right]a_f$$

flops.

Next we apply the interpolation type quadrature formula with the previously calculated coefficients (reordering the terms) using

$$(N+1)\binom{d+k-1}{d}$$

operations. About the contribution of the Lagrange interpolation polynomials note that

$$S=\sum_{j=1}^{N}\sum_{s=1}^{m}L_k(f,\xi(j,s))=\sum_{j=1}^{N}\sum_{s=1}^{m}\sum_{i=1}^{\binom{d+k-1}{d}}f\left(y^{(i)}\right)\sum_{r=1}^{\binom{d+k-1}{d}}t_{ir}\mu_r\left(\xi(j,s)\right),$$

where μ_r are all the monomials of degree less than k. Reordering the terms we obtain

$$S=\sum_{j=1}^{N}\sum_{i=1}^{\binom{d+k-1}{d}}f\left(y^{(i)}\right)\sum_{r=1}^{\binom{d+k-1}{d}}t_{ir}\sum_{s=1}^{m}\mu_r\left(\xi(j,s)\right).$$

Using the fact the the value of each monomial can be obtained using only one multiplication, once we know all the values of the monomials of lesser degree at the same point, we see that S can be calculated with less than

$$(2m-1)N\binom{d+k-1}{d}+\left(2\binom{d+k-1}{d}+2\right)\binom{d+k-1}{d}N$$

flops. The summing of all function values at the random points $\xi(s,k)$ takes mN-1 flops. Now we sum all these estimates to obtain

$$\begin{aligned}
N_{fp} &\leq 2dN\binom{d+k-1}{d}+mdN\left(b_r+2\right)+ &(4)\\
&N\left[m+\binom{d+k-1}{d}\right]a_f+(2m+1)N\binom{d+k-1}{d}+ &(5)\\
&2N\binom{d+k-1}{d}^2+(N+1)\binom{d+k-1}{d}+mN+c_1(d,k) &(6)\\
&= N\left[m+\binom{d+k-1}{d}\right]a_f+mN\left(d\left(b_r+2\right)+1\right)+ &(7)\\
&N\binom{d+k-1}{d}\left[2m+1+2d+2\binom{d+k-1}{d}+1\right]+c(d,k). &(8)
\end{aligned}$$

The theorem is proven $\diamondsuit$

Algorithm 2 (A.2)

The second algorithm is a variation of the first one, when first some k different points $z_i^{(j)} \in (0,1)$ are selected in each dimension, and then the points $x^{(r)}$ have coordinates

$$\left\{(z_1^{(j_1)}, \dots, z_d^{(j_d)}) : (j_1 + \ldots + j_d < k)\right\}.$$

In this case the interpolation polynomial is calculated in the form of Newton, namely if $w_r^{(t)} = a_r^j + (b_r^j - a_r^j) z_r^{(t)}$, then

$$L_k(f,\xi) = \sum_{j_1+\ldots+j_d<k} R(j_1, ..., j_d, 0, ..., 0) \prod_{i=1}^{d} \left(\xi_i\,(j,s) - w_i^{(1)}\right) ... \left(\xi_i\,(j,s) - w_i^{(j_i-1)}\right),$$

where

$$R(j_1, \dots, j_d, l_1, ..., l_d) = f(w_1^{j_1}, ..., w_d^{j_d}) \text{ if all } j_i = l_i,$$

and

$$R(j_1, \dots, j_i, \dots, j_d, l_1, \dots, l_i, \dots, l_d) = \tag{9}$$
$$\frac{1}{(w_i^{j_i} - w_i^{l_i})}[R(j_1, \dots, j_i, \dots, j_d, l_1, \dots, l_i + 1, \dots, l_d) - \tag{10}$$
$$R(j_1, \dots, j_i - 1, \dots, j_d, l_1, \dots, l_i, \dots, l_d)] \tag{11}$$

if $j_i > l_i$.

In this modification we have the following

Theorem 3.2 *The computational complexity of the numerical integration of a function from $\mathbf{C}_k^d$ using Algorithm A.2 is estimated by:*

$$N_{fp} \;\leq\; N\left[m + \begin{pmatrix} d+k-1 \\ d \end{pmatrix}\right] a_f + Nm\left[d(b_r + 2 + k) + 1\right] + \tag{12}$$
$$N \begin{pmatrix} d+k-1 \\ d \end{pmatrix} (2d + 1 + 2m + 1) + c\,(d,k) \tag{13}$$

where a_f and b_m*are as above.*

P r o o f. One can see that for the calculation of the divided differences in d dimensions required for the Lagrange - Newton approximation we need to apply (9) exactly

$$dN\begin{pmatrix} d+k-1 \\ d+1 \end{pmatrix}$$

times, which is less than

$$Nk\begin{pmatrix} d+k-1 \\ d \end{pmatrix}.$$

For the calculation of the sum

$$\begin{aligned} S &= \sum_{j=1}^{N}\sum_{s=1}^{m} L_k\left(f, \xi\left(j,s\right)\right) = \sum_{j=1}^{N}\sum_{s=1}^{m} \sum_{j_1+\ldots+j_d<k} R(j_1, ..., j_d, 0, ..., 0) \times & (14) \\ & \prod_{i=1}^{d}\left(\xi_i\left(j,s\right) - w_i^{(1)}\right) ... \left(\xi_i\left(j,s\right) - w_i^{(j_i-1)}\right) & (15) \end{aligned}$$

we make use of the fact that each term

$$\prod_{i=1}^{d}\left(\xi_i\left(j,s\right) - w_i^{(1)}\right) ... \left(\xi_i\left(j,s\right) - w_i^{(j_i-1)}\right)$$

can be obtained from a previously computed one through one multiplication, provided we have computed all the dk differences $\xi_i\left(j,s\right) - w_i^{(r)}$. Thus, we are able to compute the sum S with less than

$$(2m+1)\begin{pmatrix} d+k-1 \\ d \end{pmatrix} N + dkmN$$

flops.

The other estimates are done in the same way as in Theorem 3.1 to obtain

$$\begin{aligned} N_{fp} &\leq 2dN\begin{pmatrix} d+k-1 \\ d \end{pmatrix} + mdN(b_r+2) + N\left[m + \begin{pmatrix} d+k-1 \\ d \end{pmatrix}\right] a_f + & (16) \\ & (N+1)\begin{pmatrix} d+k-1 \\ d \end{pmatrix} + (2m+1)\begin{pmatrix} d+k-1 \\ d \end{pmatrix} N + dkmN + & (17) \\ & 2Nk\begin{pmatrix} d+k-1 \\ d \end{pmatrix} + mN + c_1(d,k) & (18) \\ &= N\left[m + \begin{pmatrix} d+k-1 \\ d \end{pmatrix}\right] a_f + Nm\left(d(b_r+2+k)+1\right) + & (19) \\ & N\begin{pmatrix} d+k-1 \\ d \end{pmatrix}(2d+1+2m+1) + c\left(d,k\right), & (20) \end{aligned}$$

which proves the theorem $\diamondsuit$

4 Numerical tests

In this section some numerical tests, showing the computational efficiency of the algorithms under consideration are given.

Here we present results for the following integrals:

$$
\begin{aligned}
I_1 &= \int_{\mathbf{E}^4} \frac{e^{(x_1+2x_2)} cos(x_3)}{1+x_2+x_3+x_4}\ dx_1dx_2dx_3dx_4; \\
I_2 &= \int_{\mathbf{E}^4} x_1 x_2^2 e^{x_1 x_2} \sin x_3 \cos x_4\ dx_1dx_2dx_3dx_4; \\
I_3 &= \int_{\mathbf{E}^4} e^{x_1} \sin x_2 \cos x_3 \log(1+x_4)\ dx_1dx_2dx_3dx_4; \\
I_4 &= \int_{\mathbf{E}^4} e^{x_1+x_2+x_3+x_4}\ dx_1dx_2dx_3dx_4;
\end{aligned}
$$

In the Tables 1, 2, 3, 4 the results of some numerical experiments in the case when $k = d = 4$ are presented. They are performed on ORIGIN-2000 machine using only one CPU.

n	Algorithm	Error	$err_{rel} * n^6$	CPU time, s.
10	A.1	$3.51\ 10^{-8}$	0.04	0.68
10	A.2	$3.11\ 10^{-8}$	0.03	0.53
15	A.1	$1.49\ 10^{-9}$	0.02	3.40
15	A.2	$1.90\ 10^{-9}$	0.02	2.68
20	A.1	$5.56\ 10^{-10}$	0.04	10.71
20	A.2	$5.71\ 10^{-10}$	0.04	8.47
25	A.1	$1.12\ 10^{-10}$	0.03	26.12
25	A.2	$1.84\ 10^{-10}$	0.05	20.67
30	A.1	$7.01\ 10^{-11}$	0.05	54.15
30	A.2	$5.90\ 10^{-11}$	0.04	42.86

Table 1: **Results of MC numerical integration performed on ORIGIN-2000 for I_1 (Exact value - 1.8369031187...)**

Some results are presented on Figures 1-4. The dependencies of the error on the number of points where the function values are taken when both algorithms are applied to the integral I_1 are presented on Figure 1. The same dependencies for the integral I_2 are presented on Figure 2. The results from the application of the iterated Simpson's method are provided for comparison.

n	Algorithm	Error	$err_{rel} * n^6$	CPU time, s.
10	A.1	$9.31\ 10^{-7}$	0.93	0.84
10	A.2	$7.96\ 10^{-7}$	0.80	0.71
15	A.1	$7.20\ 10^{-8}$	0.81	4.20
15	A.2	$7.69\ 10^{-8}$	0.87	3.55
20	A.1	$1.31\ 10^{-8}$	0.83	13.24
20	A.2	$1.71\ 10^{-8}$	1.09	11.69
25	A.1	$3.75\ 10^{-9}$	0.92	31.49
25	A.2	$2.97\ 10^{-9}$	0.73	27.40
30	A.1	$1.16\ 10^{-9}$	0.84	66.97
30	A.2	$9.66\ 10^{-10}$	0.70	56.80

Table 2: **Results of MC numerical integration performed on ORIGIN-2000 for I_2 (Exact value - 0.1089748630...)**

The Figure 3 shows how increases the computational time for the calculation of integral I_2 when n increases. On Figure 4 we compare the CPU time and relative error of the calculation of integral I_2 using algorithm A.2 with two different values of the parameter m - 1 and 16. One can see that setting $m = 16$ yields roughly twice better results for the same amount of CPU time.

5 Concluding remarks

- A new Monte Carlo method for calculating multi-dimensional integrals of smooth functions is presented and studied. It is proven that the method has the highest possible rate of convergence, i.e. it is an optimal superconvergent method.

- Two algorithms implementing the method are described. Estimates for the computational complexity of both algorithms (A.1 and A.2) are presented.

- The numerical examples show that both algorithms give comparable results for the same number of points where the function values are taken.

- In all our examples Algorithm A.2 is less time consuming. It offers more possibilities for use in high dimensions and for functions with high order smoothness. We demonstrated how one can achieve better results for the same computational time using carefully chosen $m > 1$.

- Both algorithms are easy to implement on parallel machines because the calculations performed for each cube are independent from that for any other. The fine granularity of the tasks and the low communication costs allow for efficient parallelization.

n	Algorithm	Error	$err_{rel} * n^6$	CPU time, s.
10	A.1	$6.28\ 10^{-8}$	0.06	1.05
10	A.2	$8.22\ 10^{-8}$	0.08	0.92
15	A.1	$7.80\ 10^{-9}$	0.09	5.28
15	A.2	$5.55\ 10^{-9}$	0.06	4.65
20	A.1	$1.39\ 10^{-9}$	0.09	16.67
20	A.1	$1.39\ 10^{-9}$	0.09	14.69
25	A.1	$2.60\ 10^{-10}$	0.06	40.66
25	A.2	$4.77\ 10^{-10}$	0.11	35.82
30	A.1	$1.20\ 10^{-10}$	0.09	84.27
30	A.2	$1.98\ 10^{-10}$	0.14	74.24

Table 3: **Results of MC numerical integration performed on ORIGIN-2000 for I_3 (Exact value - 0.2567581493).**

- It is important to know how expensive are the most important parts of the algorithms. Our measurements in the case of integral I_2 yield the following results :
  - Algorithm A.1:
    – 2 % to obtain the uniformly distributed random points;
    – 73 % to calculate all the values of f at all points;
    – 1 % to apply the interpolation type quadrature formula;
    – 18 % to calculate the Lagrange interpolation polynomial at the random points.
    – 6 % other tasks
  - Algorithm A.2:
    – 2 % to obtain the uniformly distributed random points;
    – 86 % to calculate the values of f at all points;
    – 1 % to apply the interpolation type quadrature formula;
    – 5 % to calculate the Lagrange-Newton approximation for the function f at all random points;
    – 6 % other tasks;

n	Algorithm	Error	$err_{rel} * n^6$	CPU time, s.
10	A.1	$7.00\ 10^{-8}$	0.06	0.46
10	A.2	$9.69\ 10^{-8}$	0.10	0.33
15	A.1	$3.80\ 10^{-9}$	0.04	2.27
15	A.2	$4.62\ 10^{-7}$	0.05	1.63
20	A.1	$7.77\ 10^{-10}$	0.05	7.15
20	A.2	$1.24\ 10^{-9}$	0.08	5.13
25	A.1	$3.11\ 10^{-10}$	0.08	17.44
25	A.2	$3.91\ 10^{-10}$	0.10	12.51
30	A.1	$5.99\ 10^{-11}$	0.04	36.13
30	A.2	$9.56\ 10^{-11}$	0.07	25.93

Table 4: **Results of MC numerical integration performed on ORIGIN-2000 for** I_4 **(Exact value - 8.7172116201).**

References

[Ba59] N.S. Bachvalov, *On the approximate computation of multiple integrals*, **Vestnik Moscow State University, Ser. Mat., Mech., Vol. 4** (1959), pp. 3–18.

[Ba61] N.S. Bachvalov, *Average Estimation of the Remainder Term of Quadrature Formulas*, **USSR Comput. Math. and Math. Phys., Vol. 1(1)** (1961), pp. 64–77.

[DR84] Davis and Rabinowitz, Methods of Numerical Integration (Academic Press, 1984).

[DG93] I. Dimov, T. Gurov, *Parallel Monte Carlo Algorithms for Calculation Integrals*, **Proc. WP&DP**, (K. Boyanov, Edt.), 1993, Sofia, pp. 426–434.

[DT93] I. Dimov, O. Tonev, *Monte Carlo algorithms: performance analysis for some computer architectures.* **J. of Computational and Applied Mathematics, Vol. 48** (1993), pp. 253–277.

[Du56] V. Dupach, Stochasticke pocetni metody, *Cas. pro pest. mat.* **81**, No 1 (1956) 55–68.

[Ka97] A. Karaivanova, *Adaptive Monte Carlo Methods for Numerical Integration*, **Mathematica Balkanika, Vol. 11** (1997), pp. 391–406.

[KD97] A. Karaivanova, I. Dimov *Error analysis of an Adaptive Monte Carlo Method for Numerical Integration* in: **IMACS Seminar on Monte Carlo methods and Applications**, Brussels, April, 1997.

[So73] I.M. Sobol, *Monte Carlo numerical methods*, **Nauka**, Moscow, 1973.

[So89] I.M. Sobol, *On Quadratic Formulas for Functions of Several Variables Satisfying a General Lipschitz Condition*, **USSR Comput. Math. and Math. Phys., Vol. 29(6)** (1989), pp. 936–941.

[SAK94] Bl. Sendov, A. Andreev, N. Kjurkchiev, Numerical Solution of Polynomial Equations, *Handbook of Numerical Analysis (General Editors: P.G. Ciarlet and J. L. Lions), Vol. 3)*, Solution of Equations in R^n (Part 2), (North-Holland, Amsterdam, N.Y.,1994).

[Sh88] J. Shaw, *Aspects of Numerical Integration and Summarization*, **Bayesian Statistics, Vol. 3** (1988), pp. 411–428.

[Ta92] M.D. Takev, *On Probable Error of the Monte Carlo Method for Numerical Integration*, **Mathematica Balkanica (New Series), Vol. 6** (1992), pp. 231–235.

[TN74] H. Tanaka, H. Nagata, *Quasi-Random Number Method for the Numerical Integration*, **Supplement of the Progress of Theoretical Physics, No. 56** (1974), pp. 121–131.

[NR96] E. Novak, Klaus Ritter, *High Dimensional Integration of Smooth Functions over cubes*, **Numerishche Mathematik** (1996), pp. 1–19.

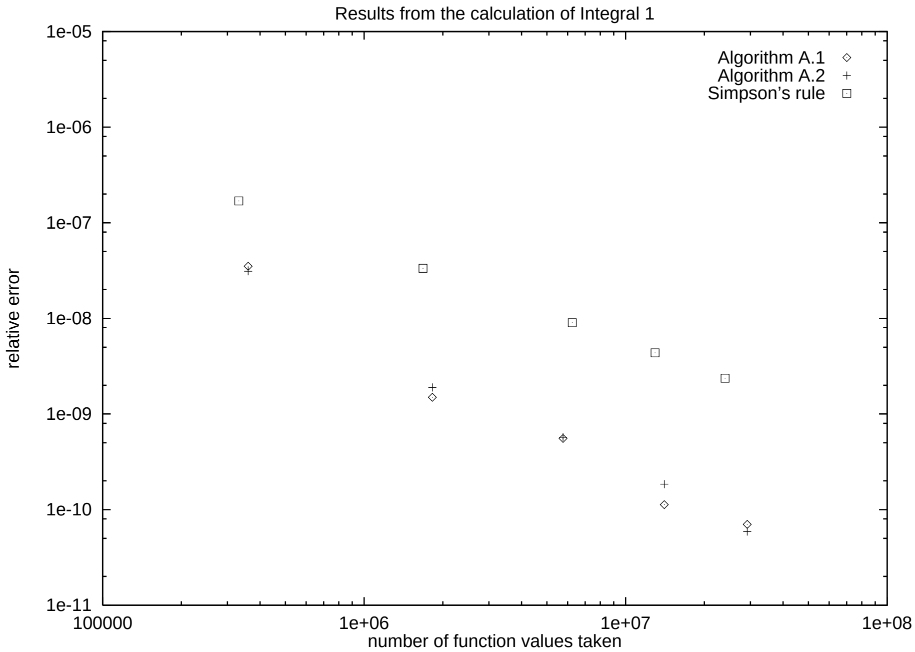

Figure 1
Errors for the integral I_1.

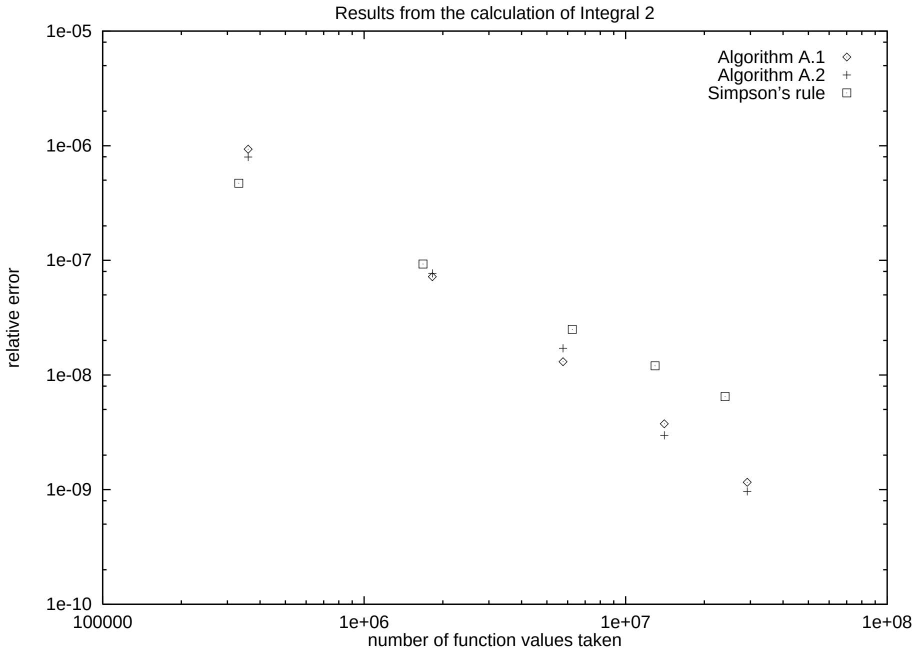

Figure 2
Errors for the integral I_2.

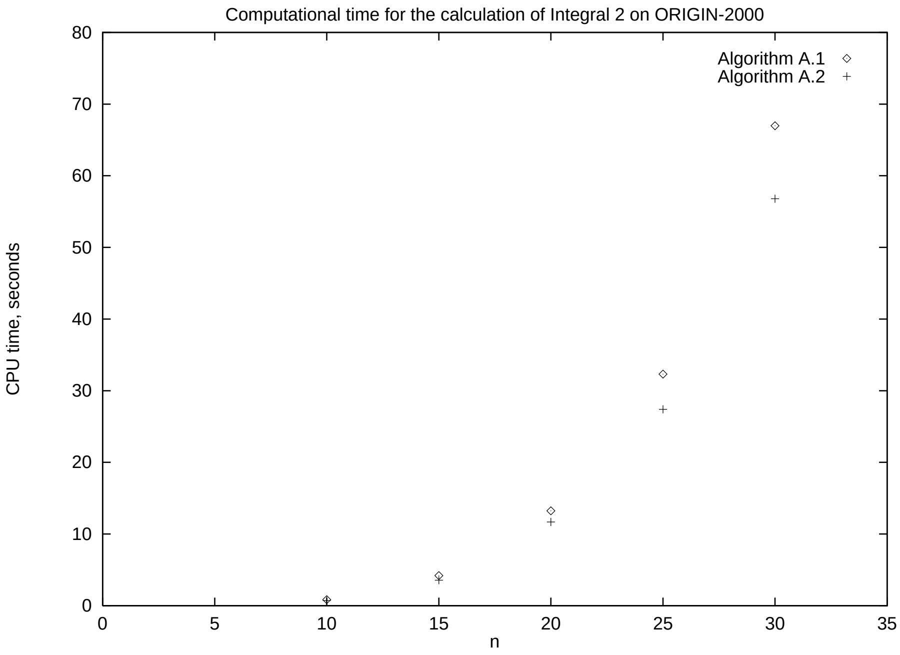

Figure 3
Computational time for I_2.

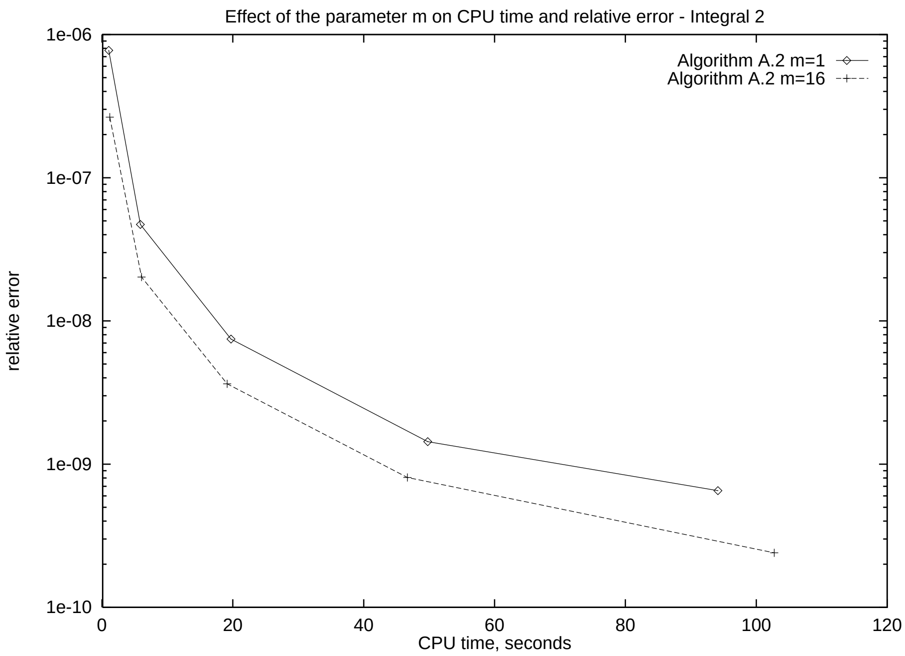

Figure 4
CPU-time and relative error for I_2 (A.2).