Monte Carlo Methods with Reduced Error

As has been shown, the probable error in Monte Carlo algorithms when no information about the smoothness of the function is used is

$$r_N = c\sqrt{\frac{D\xi}{N}}.$$

It is important for such computational schemes and random variables that a value of ξ is chosen so that the variance is as small as possible. Monte Carlo algorithms with reduced variance compared to $Plain$ Monte Carlo algorithms are usually called $efficient\ Monte\ Carlo\ algorithms$. The techniques used to achieve such a reduction are called $variance\text{-}reduction$ techniques. Let us consider several classical algorithms of this kind.

Separation of Principal Part

Consider again the integral

$$I = \int_{\Omega} f(x)p(x)dx, \tag{16}$$

where $f \in \mathbf{L}_2(\Omega, p),\ x \in \Omega \subset \mathbb{R}^d$.

Let the function $h(x) \in \mathbf{L}_2(\Omega, p)$ be *close* to $f(x)$ with respect to its $\mathbf{L}_2$ norm; i.e. $\|f - h\|_{\mathbf{L}_2} \leq \varepsilon$. Let us also suppose that the value of the integral

$$\int_{\Omega} h(x)p(x)dx = I'$$

is known.

The random variable $\theta' = f(\xi) - h(\xi) + I'$ generates the following estimate for

the integral (16)

$$\theta'_N = I' + \frac{1}{N}\sum_{i=1}^{N}[f(\xi_i) - h(\xi_i)]. \tag{17}$$

Obviously $E\theta'_N = I$ and (17) defines a Monte Carlo algorithm, which is called the $Separation\ of\ Principal\ Part$ algorithm. A possible estimate of the variancevariance of θ' is

$$D\theta' = \int_\Omega [f(x) - h(x)]^2 p(x)dx - (I - I')^2 \leq \varepsilon^2.$$

This means that the variance and the probable error will be quite small, if the function $h(x)$ is such that the integral I' can be calculated analytically. The function $h(x)$ is often chosen to be piece-wise linear function in order to compute the value of I' easily.

Integration on a Subdomain

Let us suppose that it is possible to calculate the integral analytically on $\Omega' \subset \Omega$ and

$$\int_{\Omega'} f(x)p(x)dx = I', \quad \int_{\Omega'} p(x)dx = c,$$

where $0 < c < 1$.

Then the integral (16) can be represented as

$$I = \int_{\Omega_1} f(x)p(x)dx + I',$$

where $\Omega_1 = \Omega - \Omega'$.

Let us define a random point ξ', in Ω_1, with probability density function $p_1(x') = p(x')/(1-c)$ and a random variable

$$\theta' = I' + (1-c)f(\xi'). \tag{18}$$

Obviously $E\theta' = I$. Therefore, the following approximate estimator can be used to compute I

$$\theta'_N = c + \frac{1}{N}(1+c)\sum_{i=1}^{N} f(\xi'_i),$$

where ξ'_i are independent realizations of the d-dimensional random point ξ'. The latter presentation defines the $Integration\ on\ Subdomain$ Monte Carlo algorithm.

The next theorem compares the accuracy of this Monte Carlo algorithm with the $Plain$ Monte Carlo.

Theorem 1. *If the variance $D\theta$ exists then*

$$D\theta' \leq (1-c)D\theta.$$

Proof 1. *Let us calculate the variances of both random variables θ (defined*

by (11) and θ' (defined by (18)):

$$D\theta = \int_{\Omega} f^2 p \; dx - I^2 = \int_{\Omega_1} f^2 p \; dx + \int_{\Omega'} f^2 p \; dx - I^2; \tag{19}$$

$$D\theta' = (1-c)^2 \int_{\Omega_1} f^2 p_1 \; dx - [(1-c) \int_{\Omega_1} f p_1 \; dx]^2$$

$$= (1-c) \int_{\Omega_1} f^2 p \; dx - \left(\int_{\Omega_1} f p \; dx \right)^2 . \tag{20}$$

Multiplying both sides of (19) by $(1-c)$ and subtracting the result from (20) yields

$$(1-c)D\theta - D\theta' = (1-c) \int_{\Omega'} f^2 p \; dx - (1-c)I^2 - (I - I')^2.$$

Using the nonnegative value

$$b^2 \equiv \int_{\Omega'} \left(f - \frac{I'}{c} \right)^2 p(x) dx = \int_{\Omega'} f^2 p \; dx - \frac{I'^2}{c},$$

one can obtain the following inequality

$$(1-c)D\theta - D\theta' = (1-c)b^2 + (\sqrt{c}I' - I'/\sqrt{c})^2 \geq 0$$

and the theorem is proved.

Symmetrization of the Integrand

For a one-dimensional integral

$$I_0 = \int_a^b f(x)dx$$

on a finite interval $[a, b]$ let us consider the random point ξ (uniformly distributed in this interval) and the random variable $\theta = (b-a)f(\xi)$. Since $E\theta = I_0$, the $Plain$ Monte Carlo algorithm leads to the following approximate estimate for I_0:

$$\overline{\theta}_N = \frac{b-a}{N}\sum_{i=1}^{N} f(\xi_i),$$

where ξ_i are independent realizations of ξ.

Consider the symmetric function

$$f_1(x) = \frac{1}{2}[f(x) + f(a+b-x)],$$

the integral of which over $[a, b]$ is equal to I_0. Consider also the random variable θ' defined as $\theta' = (b-a)f_1(\xi)$.

Since $E\theta' = I_0$, the following symmetrized approximate estimate of the integral may be employed:

$$\overline{\theta}_N = \frac{b-a}{2N}\sum_{i=1}^{N}[f(\xi_i) + f(a+b-\xi_i)].$$

Theorem 2. *If the partially continuous function f is monotonic in the interval $a \le x \le b$, then*

$$D\theta' \le \frac{1}{2}D\theta.$$

Proof 2. *The variances of θ and θ' may be expressed as*

$$D\theta = (b-a)\int_a^b f^2(x)dx - I_0^2, \tag{21}$$

$$2D\theta' = (b-a)\int_a^b f^2 dx + (b-a)\int_a^b f(x)f(a+b-x)dx - I_0^2. \qquad (22)$$

From (21) and (22) it follows that the assertion of the theorem is equivalent to establishing the inequality

$$(b-a)\int_a^b f(x)f(a+b-x)dx \le I_0^2. \qquad (23)$$

Without loss of generality, suppose that f *is non-decreasing,* $f(b) > f(a)$*, and introduce the function*

$$v(x) = (b-a)\int_a^x f(a+b-t)dt - (x-a)I_0,$$

which is equal to zero at the points $x = a$ *and* $x = b$*. The derivative of* v*, namely*

$$v'(x) = (b-a)f(a+b-x) - I_0$$

is monotonic and since

$$v'(a) > 0,\ v'(b) < 0, \quad \text{we see that } v(x) \geq 0$$

for $x \in [a, b]$. *Obviously,*

$$\int_a^b v(x) f'(x) dx \geq 0. \tag{24}$$

Thus integrating (24) by parts, one obtains

$$\int_a^b f(x) v'(x) dx \leq 0. \tag{25}$$

Now (23) follows by replacing the expression for $v'(x)$ *in (25). (The case of a non-increasing function* f *can be treated analogously.)*

Importance Sampling Algorithm

Consider the problem of computing the integral

$$I_0 = \int_\Omega f(x)dx, \quad x \in \Omega \subset \mathbb{R}^d.$$

Let Ω_0 be the set of points x for which $f(x) = 0$ and $\Omega_+ = \Omega - \Omega_0$.

Definition 1. *Define the probability density function* $p(x)$ *to be* $tolerant$ *to* $f(x)$*, if* $p(x) > 0$ *for* $x \in \Omega_+$ *and* $p(x) \geq 0$ *for* $x \in \Omega_0$*.*

For an arbitrary tolerant probability density function $p(x)$ for $f(x)$ in Ω let us define the random variable θ_0 in the following way:

$$\theta_0(x) = \begin{cases} \frac{f(x)}{p(x)}, & x \in \Omega_+, \\ 0, & x \in \Omega_0. \end{cases}$$

It is interesting to consider the problem of finding a tolerant density, $p(x)$, which minimizes the variance of θ_0. The existence of such a density means that the optimal Monte Carlo algorithm with minimal probability error exists.

Theorem 3. *(Kahn). The probability density function $c|f(x)|$ minimizes $D\theta_0$ and the value of the minimum variance is*

$$D\hat{\theta}_0 = \left[\int_\Omega |f(x)|dx\right]^2 - I_0^2. \tag{26}$$

Proof 3. *Let us note that the constant in the expression for $\hat{p}(x)$ is*

$$c = \left[\int_\Omega |f(x)|dx\right]^{-1},$$

because the condition for normalization of probability density must be satisfied. At the same time

$$D\theta_0 = \int_{\Omega_+} \frac{f^2(x)}{p(x)}dx - I_0^2 = D\hat{\theta}_0. \tag{27}$$

It is necessary only to prove that for other tolerant probability density functions $p(x)$ *the inequality* $D\theta_0 \geq D\hat{\theta}_0$ *holds. Indeed*

$$\left[\int_\Omega |f(x)|dx\right]^2 = \left[\int_{\Omega_+} |f|dx\right]^2 = \left[\int_{\Omega_+} |f|p^{-1/2}p^{1/2}dx\right]^2$$

Applying the Cauchy-Schwarz inequality to the last expression one gets

$$\left[\int_\Omega |f(x)|dx\right]^2 \leq \int_{\Omega_+} f^2p^{-1}dx \int_{\Omega_+} p\ dx \leq \int_{\Omega_+} \frac{f^2}{p}dx. \qquad (28)$$

Corollary 1. *If* f *does not change sign in* Ω, *then* $D\hat{\theta}_0 = 0$.

Proof 4. *The corollary is obvious and follows from the inequality (26). Indeed, let us assume that the integrand* $f(x)$ *is non-negative. Then* $\left[\int_\Omega |f(x)|dx\right] = I_0$ *and according to (26) the variance* $D\hat{\theta}_0$ *is zero. If* $f(x)$ *is non-positive, i.e.,* $f(x) \leq 0$, *then again* $\left[\int_\Omega |f(x)|dx\right] = I_0$ *and* $D\hat{\theta}_0$.

For practical algorithms this assertion allows random variables with small variances (and consequently small probable errors) to be incurred, using a higher random point probability density in subdomains of Ω, where the integrand has a large absolute value. It is intuitively clear that the use of such an approach should increase the accuracy of the algorithm.

Weight Functions Approach

Monte Carlo quadratures with weight functions are considered for the computation of

$$S(g; m) = \int g(\theta) m(\theta) d\theta,$$

where g is some function (possibly vector or matrix valued).

The unnormalized *posterior* density m is expressed as the product of two functions w and f, where w is called the *weight function* $m(\theta) = w(\theta) f(\theta)$. The weight function w is nonnegative and integrated to one; i.e., $\int w(\theta) d\theta = 1$, and it is chosen to have similar properties to m.

Most numerical integration algorithms then replace the function $m(\theta)$ by a discrete approximation in the form of

$$\hat{m}(\theta) = \begin{cases} w_i f(\theta), & \theta = \theta_i, i = 1, 2, \ldots, n, \\ 0 & \text{elsewhere,} \end{cases}$$

so that the integrals $S(g;m)$ may by estimated by

$$\hat{S}(g;m) = \sum_{i=1}^{N} w_i f(\theta_i) g(\theta_i). \tag{29}$$

Integration algorithms use the weight function w as the kernel of the approximation of the integrand

$$\begin{aligned} S(g;m) &= \int g(\theta) m(\theta) d\theta = \int g(\theta) w(\theta) f(\theta) d\theta & (30) \\ &= \int g(\theta) f(\theta) dW(\theta) = E w(g(\theta) f(\theta)). & (31) \end{aligned}$$

This suggests a Monte Carlo approach to numerical integration: generate nodes $\theta_1, \ldots, \theta_N$ independently from the distribution w and estimate $S(g;m)$ by $\hat{S}(g;m)$ in (29) with $w_i = \frac{1}{N}$. If $g(\theta)f(\theta)$ is a constant then $\hat{S}(g;m)$ will

be exact. More generally $\hat{S}(g;m)$ is unbiased and its variance will be small if $w(\theta)$ has a similar shape to $|g(\theta)m(\theta)|$. The above procedure is also known as $importance\ sampling$. Such an approach is efficient if one deals with a set of integrals with different weight functions.

Determination of the weight function can be done iteratively using $a\ posterior$ information.

Superconvergent Monte Carlo Algorithms

As was shown earlier, the probability error usually has the form of (9): $R_N = cN^{-1/2}$. The speed of convergence can be increased if an algorithm with a probability error $R_N = cN^{-1/2-\psi(d)}$ can be constructed, where c is a constant, $\psi(d) > 0$ and d is the dimension of the space. As will be shown later such algorithms exist. This class of Monte Carlo algorithms exploit the existing smoothness of the integrand. Often the exploiting of smoothness is combined with subdividing the domain of integration into a number of non-overlapping sub-domains. Each sub-domain is called $stratum$. This is the reason to call the techniques leading to superconvergent Monte Carlo algorithms $Stratified\ sampling$, or $Latin\ hypercube\ sampling$. Let us note that the $Plain$ Monte Carlo, as well as algorithms based on variance reduction techniques, do not exploit any smoothness (regularity) of the integrand. We will show how one can exploit the regularity to increase the convergence of the algorithm.

Definition 2. *Monte Carlo algorithms with a probability error*

$$R_N = cN^{-1/2-\psi(d)} \tag{32}$$

(c is a constant, $\psi(d) > 0$) are called Monte Carlo algorithms with a $superconvergent\ probable\ error$.

Error Analysis

Let us consider the problem of computing the integral

$$I = \int_{\Omega} f(x)p(x)dx,$$

where $\Omega \in \mathbb{R}^d$, $f \in \mathbf{L}_2(\Omega; p) \cap \mathbf{W}^1(\alpha; \Omega)$ and p is a probability density function, i.e. $p(x) \geq 0$ and $\int_{\Omega} p(x)dx = 1$. The class $\mathbf{W}^1(\alpha; \Omega)$ contains function $f(x)$ with continuous and bounded derivatives ($\left|\frac{\partial f}{\partial x_{(k)}}\right| \leq \alpha$ for every $k = 1, 2, \ldots, d$).

Let $\Omega \equiv \mathbf{E^d}$ be the unit cube

$$\Omega = \mathbf{E^d} = \{\mathbf{0} \leq \mathbf{x_{(i)}} < \mathbf{1}; \quad \mathbf{i = 1, 2, \ldots, d}\}.$$

Let $p(x) \equiv 1$ and consider the partition of Ω into the subdomains (stratums) $\Omega_j, j = 1, 2, \ldots, m$, of $N = m^d$ equal cubes with edge $1/m$ (evidently $p_j = 1/N$

and $d_j = \sqrt{d}/m$) so that the following conditions hold:

$$\Omega = \bigcup_{j=1}^{m} \Omega_j, \quad \Omega_i \cap \Omega_j = \emptyset, \quad i \neq j,$$

$$p_j = \int_{\Omega_j} p(x)dx \leq \frac{c_1}{N}, \tag{33}$$

and

$$d_j = \sup_{x_1, x_2 \in \Omega_j} |x_1 - x_2| \leq \frac{c_2}{N^{1/d}}, \tag{34}$$

where c_1 and c_2 are constants.

Then $I = \sum_{j=1}^{m} I_j$, where $I_j = \int_{\Omega_j} f(x)p(x)dx$ and obviously I_j is the mean of the random variable $p_j f(\xi_j)$, where ξ_j is a random point in Ω_j with probability density function $p(x)/p_j$. So it is possible to estimate I_j by the average of N_j

observations

$$\overline{\theta}_N = \frac{p_j}{N_j} \sum_{s=1}^{N_j} f(\xi_j), \quad \sum_{j=1}^{m} N_j = N,$$

and I by $\theta_N^* = \sum_{j=1}^{m} \overline{\theta}_{n_j}$.

Theorem 4. *Let $N_j = 1$ for $j = 1, \ldots, m$ (so that $m = N$). The function f has continuous and bounded derivatives ($\left|\frac{\partial f}{\partial x_{(k)}}\right| \leq \alpha$ for every $k = 1, 2, \ldots, d$) and let there exist constants c_1, c_2 such that conditions (33) and (34) hold.*

Then for the variance of θ^* the following relation is fulfilled

$$D\theta_N^* = (dc_1c_2\alpha)^2 N^{-1-2/N}.$$

Using the Tchebychev's inequality it is possible to obtain

$$R_N = \sqrt{2} dc_1c_2\alpha N^{-1/2-1/d}. \tag{35}$$

The Monte Carlo algorithm constructed above has a superconvergent probability error. Comparing (35) with (32) one can conclude that $\psi(d) = \frac{1}{d}$. We can try to relax a little bit the conditions of the last theorem because they are too strong. So, the following problem may be considered: *Is it possible to obtain the same result for the convergence of the algorithm but for functions that are only continuous?*

Let us consider the problem in $\mathbb{R}$. Let $[a, b]$ be partitioned into n subintervals $[x_{j-1}, x_j]$ and let $d_j = |x_j - x_{j-1}|$. Then if ξ is a random point in $[x_{j-1}, x_j]$ with probability density function $p(x)/p_j$, where $p_j = \int_{x_{j-1}}^{x_j} p(x)dx$, the probable error of the estimator θ_N^* is given by the following:

Theorem 5.

Let f be continuous in $[a, b]$ and let there exist positive constant c_1, c_2, c_3

satisfying $p_j \leq c_1/N$, $c_3 \leq d_j \leq c_2/N$ *for* $j = 1, 2, \ldots, N$. *Then*

$$r_N \leq 4\sqrt{2}\frac{c_1 c_2}{c_3}\tau\left(f; \frac{3}{2}d\right)_{L_2} N^{-3/2},$$

where $d = \max_j d_j$ *and* $\tau(f;\delta)_{L_2}$ *is the averaged modulus of smoothness, i.e.*

$$\tau(f;\delta)_{\mathbf{L}_2} = \| \omega(f, \bullet; \delta) \|_{\mathbf{L}_2} = \left(\int_a^b (\omega(f, x; \delta))^q dx\right)^{1/q}, \ 1 \leq q \leq \infty,$$

$\delta \in [0, (b-a)]$ *and*

$$\omega(f, x; \delta) = sup\{|\Delta_h f(t)| : t, t+h \in [x - \delta/2, x + \delta/2] \cap [a, b]\}.$$

where Δ_h *is the restriction operator.*

In $\mathbb{R}^d$ the following theorem holds:

Theorem 6. *(Dimov, Tonev)*

Let f be continuous in $\Omega \subset \mathbb{R}^d$ and let there exist positive constants c_1, c_2, c_3 such that $p_j \leq c_1/N$, $d_j \leq c_2 N^{1/d}$ and $S_j(\bullet, c_3) \subset \Omega_j$, $j = 1, 2, \ldots, N$, where $S_j(\bullet, c_3)$ is a sphere with radius c_3. Then

$$r_N \leq 4\sqrt{2}\frac{c_1 c_2}{c_3}\tau(f; d)_{\mathbf{L}_2} N^{-1/2-1/d}.$$

Let us note that the best quadrature formula with fixed nodes in $\mathbb{R}$ in the sense of Nikolskiy for the class of functions $\mathbf{W}^{(1)}(l; [a, b])$ is the rectangular rule with equidistant nodes, for which the error is approximately equal to c/N. For the Monte Carlo algorithm given by Theorem 6 when $N_j = 1$ the rate of convergence is improved by an order of $1/2$. This is an essential improvement. At the same time, the estimate given in Theorem 5 for the rate of convergence attains the lower bound estimate obtained by Bakhvalov for the error of an arbitrary random quadrature formula for the class of continuous functions in an interval $[a, b]$. Some further developments in this direction will be presented in Chapter .

II. Optimal Monte Carlo Method for Multidimensional Integrals of Smooth Functions

An optimal Monte Carlo method for numerical integration of multidimensional 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 $O\left(N^{-\frac{1}{2}-\frac{k}{d}}\right)$. 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.

Here 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 $\mathbf{W}^k(\|f\|; \mathbf{E}^d)$ (sometimes abbreviated to $\mathbf{W}^k$) of real functions f defined over the unit cube $\mathbf{E^d} = [\mathbf{0}, \mathbf{1})^\mathbf{d}$, possessing all the partial derivatives

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

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

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

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

$$I(f) = \sum_{i=1}^{N} c_i f(x^{(i)}), \tag{36}$$

where $x^{(i)} \equiv \left(x_1^{(i)}, \dots, x_d^{(i)} \right) \in \mathbf{E^d}$, $\mathbf{i = 1, \dots, N}$, are the nodes and c_i, $i = 1, \dots, N$ are the weights. If $x^{(i)}$ and c_i are real values, the formula (36) 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 $\mathbb{R}$ then (36) defines a Monte Carlo quadrature formula.

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

Theorem 7. *(Bakhvalov) There exists a constant $c(d,k)$ such that for every quadrature formula $I(f)$ that is fully deterministic and uses the function values at N points there exists a function $f \in \mathbf{W}^k$ such that*

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

Theorem 8. *(Bakhvalov) There exists a constant $c(d,k)$ such that for every quadrature formula $I(f)$ that involves random variables and uses the function values at N points there exists a function $f \in \mathbf{W}^k$ 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, that 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 Definition 2 given before, hey have a 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, when $k \geq 3$ such methods become much more sophisticated. The first optimal stochastic method was proposed by Mrs. Dupach 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 each subdomain. This idea was largely used for creation Monte Carlo methods with high rate of convergence.

There exist also the so-called $adaptive$ Monte Carlo methods proposed by Lautrup, 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. 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 $k = 1$.

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 each cube K_j we calculate the coordinates of $\binom{d+k-1}{d}$ 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.

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). We have shown that the computational complexity could be estimated as a product of $t\sigma^2(\theta)$, where t is the time (number of operations) needed to calculate one value of the random variable θ, whose mathematical expectation is equal to the exact value of the integral under consideration and $\sigma^2(\theta)$ is the variance. Here we do not use this presentation and, instead, 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), such as $speedup$ and $parallel\ efficiency$. 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.

Description of the Method and Theoretical Estimates

Definition 3. *Given a Monte Carlo integration formula for the functions in the space* $\mathbf{W}^k$ *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 $\binom{d+k-1}{d}$ points in $\mathbf{E^d}$ in the following way:

The $\binom{d+k-1}{d}$ 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

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

with $b_i^j - a_i^j = \frac{1}{n}$ for all $i = 1, \ldots, d$. Now in each cube K_j we calculate the coordinates of $\binom{d+k-1}{d}$ 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

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

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

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

We prove the following

Theorem 9. *The quadrature formula constructed above satisfies*

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

and

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

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

Proof 5. *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

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

$$=\quad \frac{1}{m}D\left\{\frac{1}{n^d}\left[f(\xi(j,1))-L_k(f,\xi(j,1))\right]\right\}.$$

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*

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

and therefore

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

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

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

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

and therefore ($N = n^d$)

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

The application of the Tchebychev's inequality yields

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

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

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. For Quasi-Monte Carlo integration of smooth functions an approach with Tchebychev polynomial approximation is developed. We (Dimov, Atanasov) show that the proposed technique allows one to formulate optimal algorithms in the Hölder class of functions $\mathbf{H}_{\lambda}^{k}(\alpha, \mathbf{E}^{d}), (0 < \lambda \leq 1)$. The class $\mathbf{H}_{\lambda}^{k}(\alpha, \mathbf{E}^{d}), (0 < \lambda \leq 1)$ is defined as functions from $\mathbf{C}^{k}$, which derivatives of order k satisfy Hölder condition with a parameter λ:

$$\begin{aligned} \mathbf{H}_{\lambda}^{k}(\alpha, \mathbf{E}^{d}) &\equiv \left\{ f \in \mathbf{C}^{k} : \left| D^{k} f(y_1, \ldots, y_d) - D^{k} f(z_1, \ldots, z_d) \right| \right. \\ &\leq \left. \alpha \sum_{j=1}^{d} |y_j - z_j|^{\lambda} \right\}. \end{aligned} \tag{37}$$

For the class $\mathbf{H}_{\lambda}^{k}(\alpha, \mathbf{E}^{d})$ we prove the following theorem:

Theorem 10. *The cubature formula constructed above satisfies*

$$r_N(f, k+\lambda, d, m) \leq c^{'}(d, k+\lambda)\frac{1}{m}\alpha N^{-\frac{1}{2}-\frac{k+\lambda}{d}}$$

and

$$\left(E\left(\int_{\mathbf{E}^d} f(x)dx - I(f)\right)^2\right)^{1/2} \leq c^{''}(d, k+\lambda)\frac{1}{m}\alpha N^{-\frac{1}{2}-\frac{k+\lambda}{d}},$$

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

The above theorem shows that the convergence of the method can be improved by adding a parameter λ to the factor of smoothness k if the integrand belongs to the Hölder class of functions $\mathbf{H}_{\lambda}^{k}(\alpha, \mathbf{E}^d), (0 < \lambda \leq 1)$.

Estimates of the Computational Complexity

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 $\mathbf{W}^k$. Let us order all the monomials of d variables and degree less than $k - \mu_1, \ldots, \mu_t$. Note that there are exactly $\binom{d+k-1}{d}$ 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 $\mathbf{W}^k$. 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 count as $O(1)$ in our estimates for the number of flops used to calculate the integral of a certain function from $\mathbf{W}^k$. These calculations could be considered as $preprocessing$. We prove the following

Theorem 11. *The computational complexity of the numerical integration of a function from $\mathbf{W}^k$ using Algorithm A.1 is estimated by:*

$$\begin{aligned} N_{fp} \quad \leq \quad & N\left[m+\binom{d+k-1}{d}\right]a_f + mN\left[d\left(b_r+2\right)+1\right]+ \\ & N\binom{d+k-1}{d}\left[2m+1+2d+2\binom{d+k-1}{d}+1\right]+c(d,k) \end{aligned} \tag{38}$$

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.

Proof 6. *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*

$$2dN\binom{d+k-1}{d}$$

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*

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

The theorem is proven.

*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

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

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

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

where

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

and

$$\begin{aligned} & R(j_1,\ldots,j_i,\ldots,j_d,l_1,\ldots,l_i,\ldots,l_d) \\ = & \frac{1}{(w_i^{j_i}-w_i^{l_i})}[R(j_1,\ldots,j_i,\ldots,j_d,l_1,\ldots,l_i+1,\ldots,l_d) \\ - & R(j_1,\ldots,j_i-1,\ldots,j_d,l_1,\ldots,l_i,\ldots,l_d)] \end{aligned}$$

if $j_i > l_i$.

In this modification we have the following

Theorem 12. *The computational complexity of the numerical integration of a function from* $\mathbf{W}^k$ *using Algorithm A.2 is estimated by:*

$$\begin{aligned} N_{fp} \quad \leq \quad & N\left[m+\binom{d+k-1}{d}\right]a_f + Nm\left[d(b_r+2+k)+1\right] \\ + \quad & N\binom{d+k-1}{d}(2d+1+2m+1)+c\left(d,k\right), \end{aligned}$$

where a_f and b_m are as above.

Proof 7. *One can see that for the calculation of the divided differences in d dimensions required for the Lagrange - Newton approximation we need to apply (39) exactly*

$$dN\binom{d+k-1}{d+1}$$

times, which is less than

$$Nk\binom{d+k-1}{d}.$$

For the calculation of the sum

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

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*

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

flops.

The other estimates are done in the same way as in Theorem 11 to obtain

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

which proves the theorem.

Numerical Tests

Numerical tests, showing the computational efficiency of the algorithms under consideration are given.

Here we present results for the following integrals:

$$I_1 = \int_{\mathbf{E}^4} \frac{e^{(x_1+2x_2)}\cos(x_3)}{1+x_2+x_3+x_4}\, dx_1dx_2dx_3dx_4;$$

$$I_2 = \int_{\mathbf{E}^4} x_1x_2^2e^{x_1x_2}\sin x_3\cos x_4\ dx_1dx_2dx_3dx_4;$$

$$I_3 = \int_{\mathbf{E}^4} e^{x_1}\sin x_2\cos x_3\log(1+x_4)\ dx_1dx_2dx_3dx_4;$$

$$I_4 = \int_{\mathbf{E}^4} e^{x_1+x_2+x_3+x_4}\ dx_1dx_2dx_3dx_4;$$

In the Tables 1 to 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...)**

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...)**

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).**

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).**

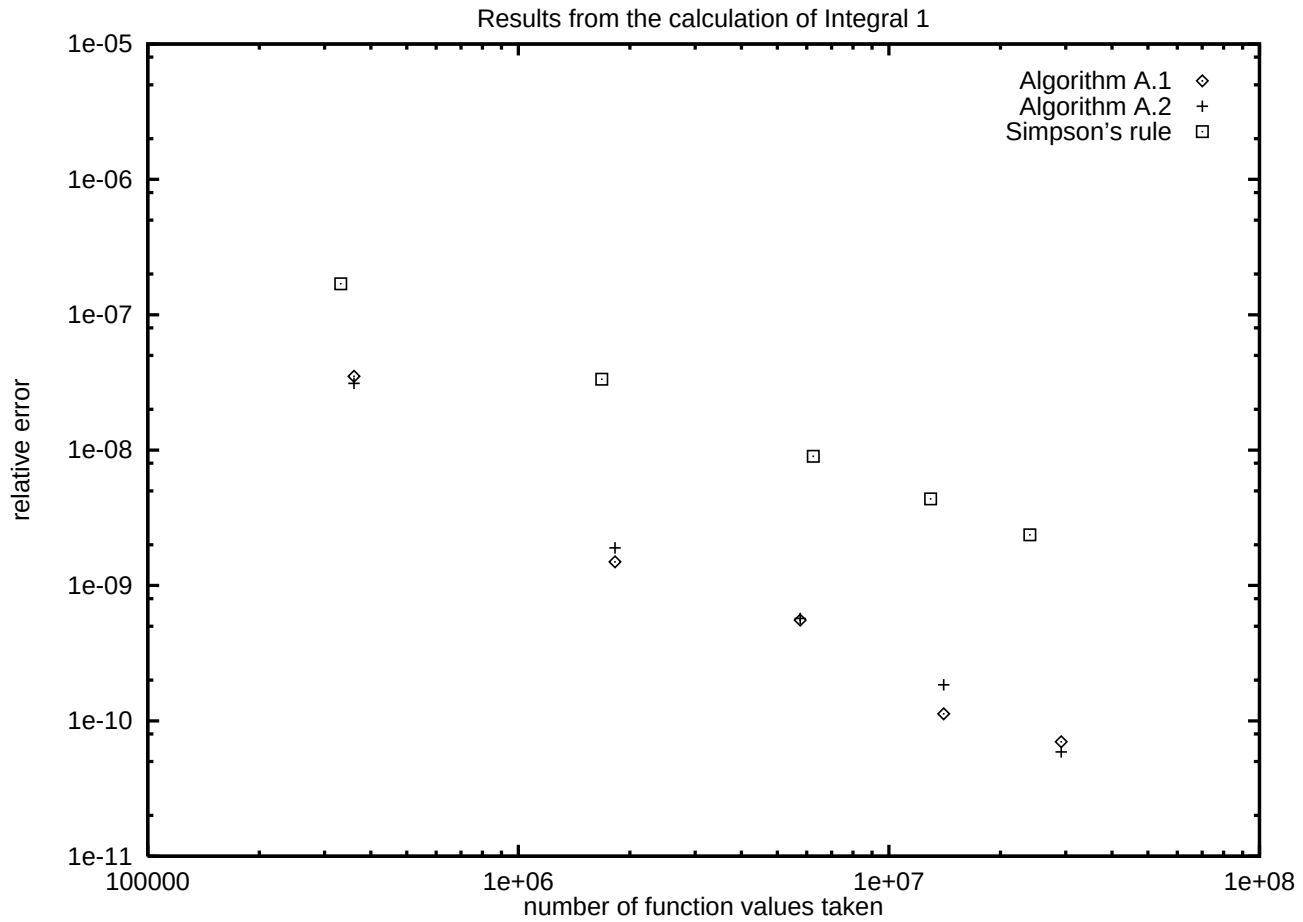

Figure 1: Errors for the integral I_1.

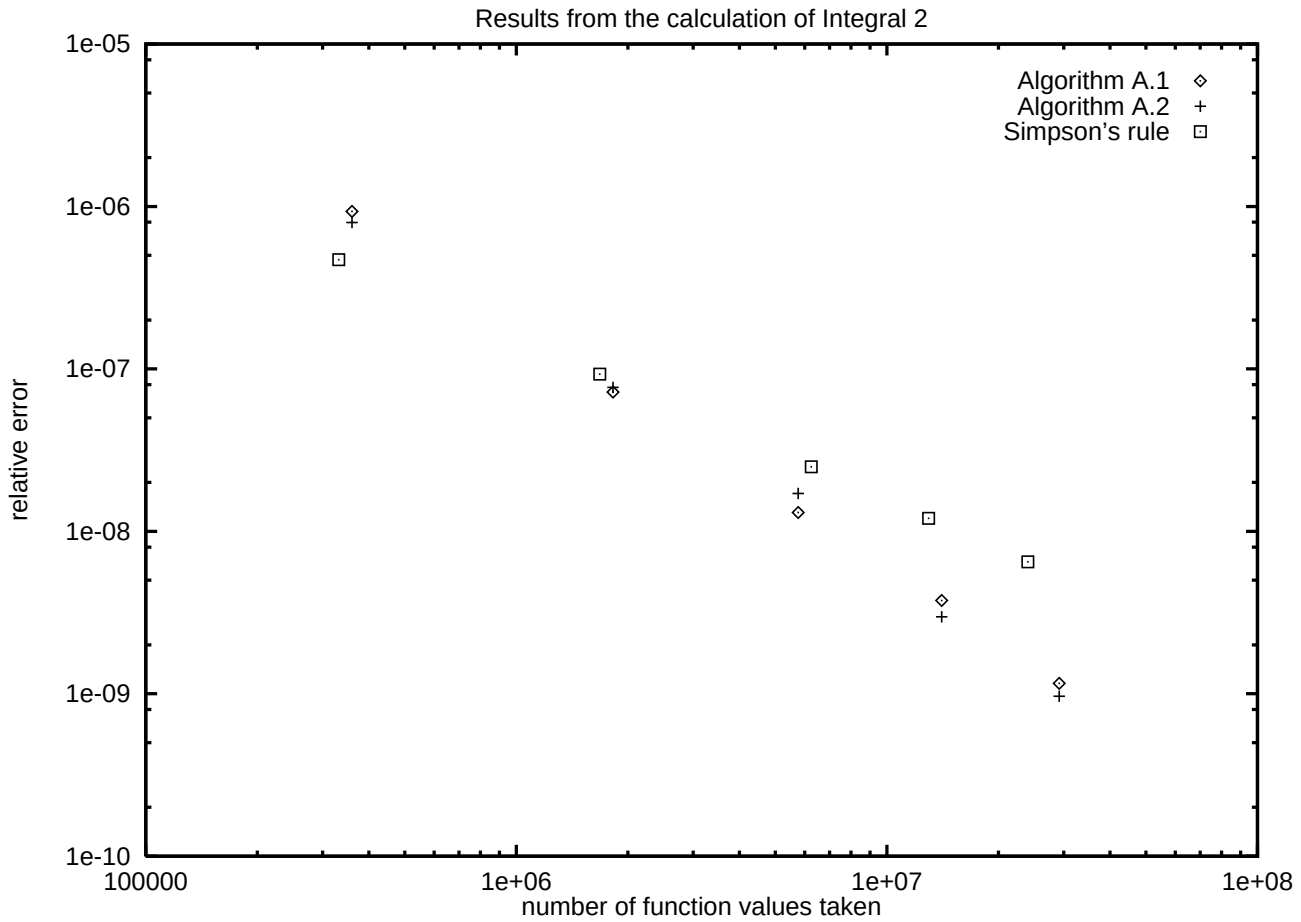

Figure 2: Errors for the integral I_2.

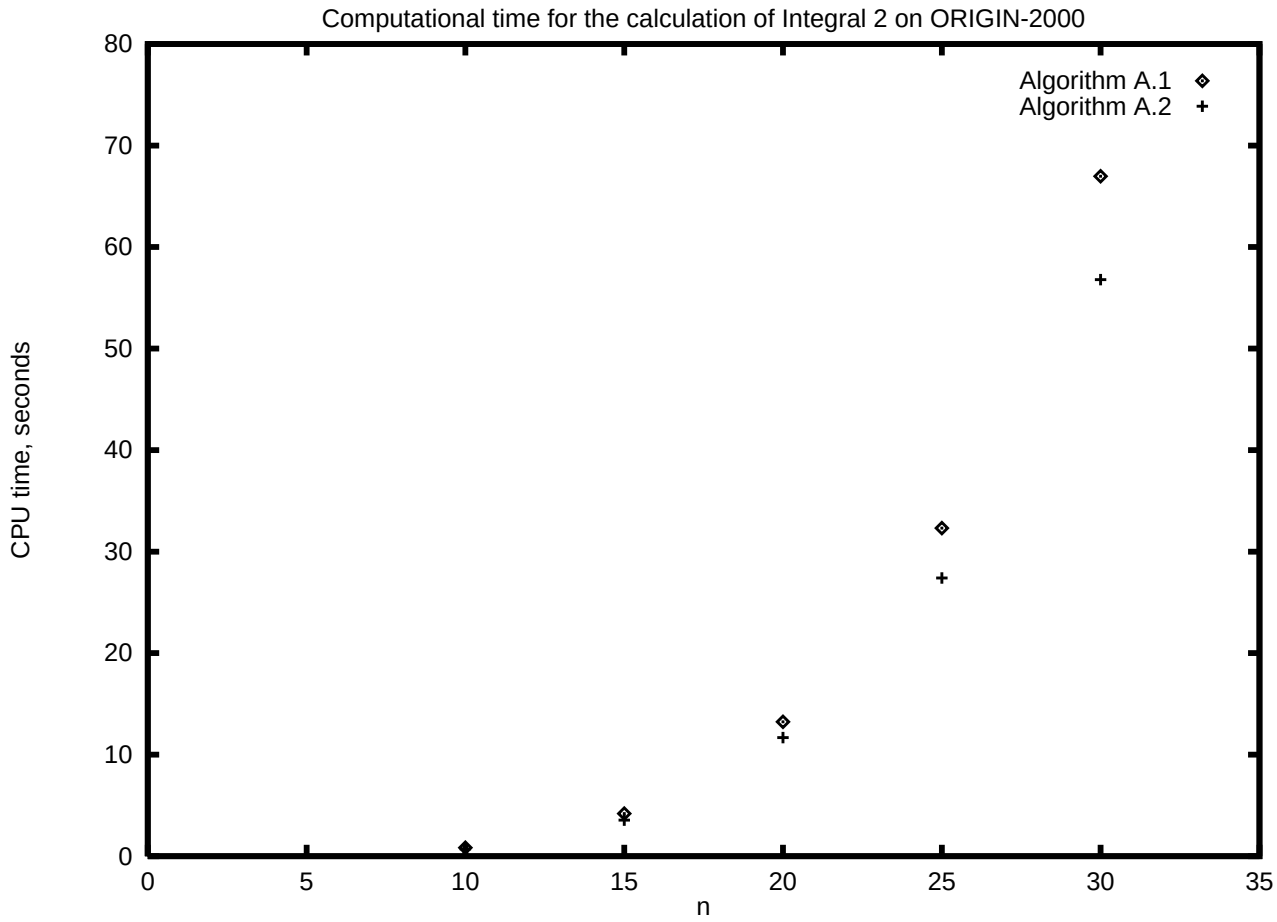

Figure 3: Computational time for I_2.

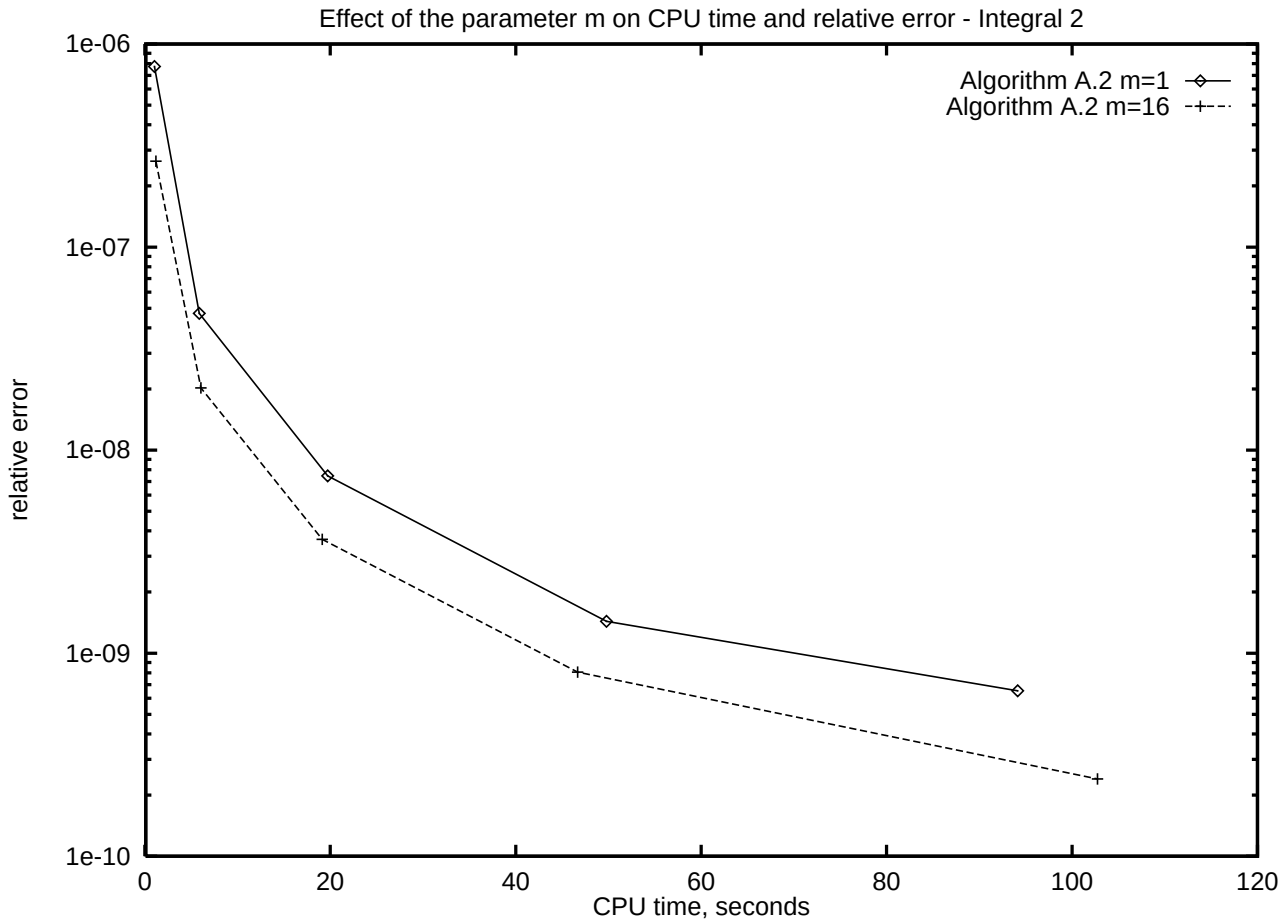

Figure 4: CPU-time and relative error for I_2 (A.2).

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.

The Figure 3 shows how increases the computational time for the calculation of integral I_2 when $n = N^{1/d}$ 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.

Concluding Remarks

- A Monte Carlo method for calculating multidimensional 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 quicker. 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.

- 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.