Monte Carlo methods for multidimensional integration for European option pricing

Conference Paper · October 2016

DOI: 10.1063/1.4965003

2 authors:

Venelin Todorov
Bulgarian Academy of Sciences
30 PUBLICATIONS **35** CITATIONS

SEE PROFILE

Ivan Dimov
Bulgarian Academy of Sciences
255 PUBLICATIONS **1,873** CITATIONS

SEE PROFILE

Monte Carlo methods for multidimensional integration for European option pricing

V. Todorov and I. T. Dimov

Citation: AIP Conference Proceedings **1773**, 100009 (2016); doi: 10.1063/1.4965003
View online: http://dx.doi.org/10.1063/1.4965003
View Table of Contents: http://scitation.aip.org/content/aip/proceeding/aipcp/1773?ver=pdfcov
Published by the AIP Publishing

Monte Carlo Methods for Multidimensional Integration for European Option Pricing

V. Todorov[a] and I.T. Dimov[b]

Department of Parallel Algorithms, Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, Acad. G. Bonchev str. 25 A, 1113 Sofia, Bulgaria

[a]Corresponding author: venelinltodorov@gmail.bg
[b]ivdimov@bas.bg

Abstract. In this paper, we illustrate examples of highly accurate Monte Carlo and quasi-Monte Carlo methods for multiple integrals related to the evaluation of European style options. The idea is that the value of the option is formulated in terms of the expectation of some random variable; then the average of independent samples of this random variable is used to estimate the value of the option. First we obtain an integral representation for the value of the option using the risk neutral valuation formula. Then with an appropriations change of the constants we obtain a multidimensional integral over the unit hypercube of the corresponding dimensionality. Then we compare a specific type of lattice rules over one of the best low discrepancy sequence of Sobol for numerical integration. Quasi-Monte Carlo methods are compared with Adaptive and Crude Monte Carlo techniques for solving the problem. The four approaches are completely different thus it is a question of interest to know which one of them outperforms the other for evaluation multidimensional integrals in finance. Some of the advantages and disadvantages of the developed algorithms are discussed.

INTRODUCTION

Nowadays Monte Carlo method has become a popular computational device for problems in finance. The finance discipline has become more sophisticated and more quantitative in the last two decades. New approaches have been developed that outperform standard Monte Carlo in terms of numerical efficiency. It has been found that there can be efficiency gains in using deterministic sequences rather than the random sequences which are a feature of standard Monte Carlo [17]. These deterministic sequences are carefully selected so that they are well dispersed throughout the region of integration. Sequences with this property are known as low discrepancy sequences. These sequences are often more efficient than standard Monte Carlo in evaluating high dimensional integrals if the integrand is sufficiently regular and for many finance applications this is the case. Applications of low discrepancy sequences to finance problems have been discussed by Boyle, Broadie and Glasserman (1997), Cashisch, Morokoff and Owen (1997), Joy, Boyle and Tan (1996), Ninomiya and Tezuka (1996), Tan and Boyle (2000) and Paskov and Traub (1995). The Monte Carlo method has proven to a very useful tool for numerical analysis, particularly when the number of dimensions ranging from medium to large.

The pricing of options is a very important problem encountered in financial markets today. The famous Black-Scholes model provides explicit closed form solutions for the values of European style call and put options. Options have been widely traded since the creation of an organized exchange in 1973. Much of the focus to-date has been on high-dimensional problems since these are more challenging from a computational viewpoint. However, it is also of interest to examine low to medium dimension problems. Low to medium sized problems are of practical interest since there are popular contracts whose value depends on a small to medium number of variables. Here are a few examples [4]:

Options whose payoff depends on the relative performance of two underlying assets. A particular version of this option known as a spread option popular in the energy industry.

Basket options where the payoff depends on the ending values of a number of assets such as different common stocks or stock market indices. The payoff could be based on the average, the maximum or the minimum of the asset

Application of Mathematics in Technical and Natural Sciences
AIP Conf. Proc. 1773, 100009-1–100009-13; doi: 10.1063/1.4965003
Published by AIP Publishing. 978-0-7354-1431-0/$30.00

prices.

Path dependent options where the payoff is a function of the asset price at a number of discrete monitoring points along the path. In the case of Asian options the payoff is based on the average of these points. In the case of lookback options the payoff is based on the largest (or smallest) value recorded at one of these monitoring points. The dimensions of the problem are directly related to the number of discrete points in the path.

Monte Carlo and/or quasi-Monte Carlo methods can be directly applied to finance problems involving multidimensional integrals. For example, Paskov uses a quasi-Monte Carlo sequence the Sobol sequence - to find the present value of securities which involve up to 360 dimensional integrals; see [29]. Besides the applications in option pricing, Monte Carlo methods have also been widely applied to other financial problems [3, 8, 19, 21, 23, 25].

The basic definitions are taken form [7, 9, 15, 26, 35].

A European call option is a contract such that the owner may (without obligation) buy some prescribed asset (called the underlying) S at a prescribed time (expiry date) T at a prescribed price (exercise or strike price) E.

A European put option is the same as a call option, except that "buy" is replaced by "sell."

A look-back option is an option whose payoff depends not only on the asset price at expiry, but also on the maximum or minimum of the asset price over some time prior to expiry. Risk neutrality is the characteristic ascribed to an investor who is indifferent with respect to risk. A rigorous definition may be given (see [6] or [16]) based on an order relationship on a space of random variables on an appropriately defined probability space. However, the most important use of this notion for us is in its application to the risk-neutral evaluation formula.

The risk-free interest rate r is an idealized interest rate, usually taken to be that of an appropriate Treasury Bond.

The Wiener process (also called Brownian motion) dX is a special type of Markov stochastic process with the following properties: $dX \sim N(0, \sqrt{(dt)})$, where $N(\mu, \sigma)$ is the normal distribution with mean μ and variance σ^2.

MULTIDIMENSIONAL INTEGRALS RELATED TO EVALUATION OF EUROPEAN STYLE OPTIONS

One of the basic problems in option pricing is: given the current price of an asset S, the strike price E, the time to expiry T, the risk-free interest rate r, and the equation that is assumed to control the behavior of S as a function of time t:

$$dS = \mu S\, dt + \sigma S\, dX, \tag{1}$$

where dX is a Wiener process, μ (a measure of the average rate of growth of the asset price) is the drift rate and σ is the volatility of the asset (characterizing fluctuations in the price S), how may one determine a "fair" current value $V(S, t)$ of the option? The well-known Black-Scholes model for a European call option can be described ([2] or [35]) by the following (diffusion-type) partial differential equation for this value:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0, \tag{2}$$

with final condition

$$V(S, T) = \max(S - E, 0),$$

and boundary conditions

$$V(0, t) = 0, V(S, t) \sim S, S \to \infty.$$

The European put option satisfies the same equation as (2), but with final condition

$$V(S, T) = \max(E - S, 0),$$

and boundary conditions

$$V(0, t) = Ee^{-r(T-t)}, V(S, t) \sim 0, S \to \infty.$$

In both cases, there are explicit closed form solutions. For the call option, the solution is

$$V(S, t) = C(S, t) = SN(d1) - Ee^{-r(T-t)}N(d2),$$

with

$$d_1 = \frac{\ln(\frac{S}{E}) + (r + \frac{\sigma^2}{2})(T - t)}{\sigma\sqrt{(T - t)}}$$

and

$$d_2 = \frac{\ln(\frac{S}{E}) + (r - \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{(T-t)}}$$

and $N(z)$ is the cumulative distribution function of the standard normal distribution. For the put option,

$$V(S,t) = P(S,t) = Ee^{-r(T-t)}N(-d2) - SN(-d1),$$

with the same $d1, d2$, and $N(z)$.

Monte Carlo methods can be very useful in such cases if the solution (*i.e.*, the value, V) can be expressed as the expectation of some random variable(s). This is made possible by the risk-neutral valuation formula for the European option [6]:

$$V(S,t) = E(e^{r(Tt)}h(S(T)) \mid S(t) = S, \mu = r), \tag{3}$$

where $E(.)$ is the expectation, $h(S)$ is the payoff function, $h(S) = \max(S - E, 0)$ for a call option and $h(S) = \max(E - S, 0)$ for a put option.

We follow the idea of Lai and Spanier in [26]. Consider an European option whose payoff depends on $k > 1$ assets with prices $S_i, i = 1, ..., k$. Each asset follows the random walk

$$dS_i = \mu_i S_i dt + \sigma_i S_i dX_i,$$

where σ_i is the annualized standard deviation for the i-th asset and dX_i is Brownian motion. Suppose at expiry time T that the payoff is given by $h(S_1^{'}, \ldots, S_k^{'})$ (where $S^{'}$ denotes the value of the i-th asset at expiry). Then the current value, V, of the option (assuming risk neutrality) will be

$$V = e^{-r(T-t)}(2\pi(T-t))^{-k/2}(\det\Sigma)^{-1/2}(\sigma_1, \ldots, \sigma_k)^{-1} \int_0^\infty \ldots \int_0^\infty \frac{h(S_1^{'}, \ldots, S_k^{'})}{S_1^{'}, \ldots, S_k^{'}} \exp(-0.5\alpha^T\Sigma^{-1}\alpha) dS_1^{'}, \ldots, dS_k^{'}, \tag{4}$$

where

$$\alpha_i = (\sigma_i(T-t)^{1/2})^{-1}\left(\log\frac{S_i^{'}}{S_i} - (r - \frac{\sigma^2}{2})(T-t)\right),$$

r is the risk-free interest rate and Σ is the covariance matrix,where the (i, j) entry is the covariance of dX_i and dX_j for the k assets. The infinite domain of integration can be mapped into the k-dimensional unit hypercube in a variety of ways. For example, $\frac{2}{\pi}\arctan(x)$ maps $(0, \infty)$ to $(0, 1)$. Such a mapping transforms the problem to one in which an integral $\int_0^1 \ldots \int_0^1 g(x_1, \ldots, x_k)dx_1 \ldots dx_k$ over the hypercube is sought. When g is the exponent function, with appropriate choices of the constants involved in the equation for the value of the option (4) according to [26] we obtain k dimensional integral $\int\limits_{[0,1]^k} \exp(x_1, \ldots, x_k)dx_1 \ldots dx_k$. We compare four very effective and highly accurate techniques for numerical integration for solving the last integral.

MONTE CARLO ALGORITHMS FOR NUMERICAL INTEGRATION

Lattice rules are based on the use of deterministic sequences rather than random sequences. They are a special type of so-called low discrepancy sequences. It is known that as long as the integral is sufficiently regular, lattice rules generally outperform not only basic Monte Carlo, but also other types of low discrepancy sequences. It is well known that Sobol algorithm has some advantageous over the other low discrepancy sequences such as Halton as it is shown its superiority for high dimensional integrals up to 360 dimensions by Paskov [29]. The monographs of Sloan and Kachoyan [31], Niederreiter [27], Hua and Wang [22] and Sloan and Joe [30] provide comprehensive expositions of the theory of integration lattices. We implemented a specific lattice rule and compared its performance with an implementation of Sobol, Crude and the Adaptive method over integrals of smooth functions.

Crude Monte Carlo

First we will demonstrate the power of the plain Monte Carlo over the deterministic methods. Suppose $f(x)$ is a continuous function and let a quadrature formula of Newton or Gauss type be used for calculating the integrals. Consider an example with $d = 20$. We generate a grid in the 20-dimensional domain and take the sum of the function values at the grid points. Let a grid be chosen with 20 nodes on the each of the coordinate axes in the 20-dimensional cube $G = [0, 1]^{20}$. In this case we have to compute about 10^{20} values of the function $f(x)$. Suppose a time of $10^{-7}s$ is necessary for calculating one value of the function. Therefore, a time of order 10^{13}s will be necessary for evaluating the integral (remember that 1 year = 31536×10^3s, and that there has been less than 9×10^{10}s since the birth of Pythagoras). If the formula of rectangles is applied then the error in the approximate integral calculation is

$$\epsilon \leq cMh^3,$$

where $h = 0.1$ is the mesh size, c is constant independent of h and M is the maximal value of the second derivative.

Consider a plain Monte Carlo algorithm for this problem with a probable error of the same order. The algorithm itself consists of generating N pseudo random values (points) (PRV) in G; in calculating the values of $f(x)$ at these points; and averaging the computed values of the function. For each uniformly distributed random point we have to generate 20 random numbers uniformly distributed in $[0, 1]$. The probable error is:

$$\epsilon \leq 0.6745\sigma(\theta)\frac{1}{\sqrt{N}}, \tag{5}$$

where $\sigma(\theta)$ is the standard deviation of random variable θ for which $E\theta = \int_G f(x)p(x)dx$ and N is the number of realizations of the random variable. The probable error is estimated in [12]:

$$\epsilon \leq 0.6745\|f\|_{L_2}\frac{1}{\sqrt{N}}. \tag{6}$$

From above equations we conclude that

$$N \approx (\frac{0.6745\|f\|_{L_2}}{cM})^2 \times h^{-6}. \tag{7}$$

Suppose that the expression in front of h^{-6} is of order 1. Since $h = 0.1$, we have $N \approx 10^6$; hence, it will be necessary to generate $20 \times 10^6 = 2 \times 10^7$ PRV. Usually, two operations are sufficient to generate a single PRV. Suppose that the time required to generate one PRV is the same as that for calculating the value of the function at one point in the domain. Therefore, in order to solve the problem with the same accuracy, a time of

$$2 \times 10^7 \times 2 \times 10^{-7} \approx 4s$$

will be necessary. The advantage of the Monte Carlo algorithms to solve such problems is obvious, in the case of 20 dimensional integral it is 2.5×10^{12} times faster than the deterministic one.

Sobol sequence

The Crude Monte Carlo method has rate of convergence $O(N^{-1/2})$ which is independent of the dimension of the integral, and that is why Monte Carlo integration is the only practical method for many high-dimensional problems [10, 11]. Much of the efforts to improve Monte Carlo are in construction of variance reduction methods which speed up the computation or to use quasi-random sequences. A quasi-random or low discrepancy sequence, such as the Faure, Halton, Hammersley, Niederreiter or Sobol sequences, is "less random" than a pseudo-random number sequence, but more useful for such tasks as approximation of integrals in higher dimensions, and in global optimization. This is because low discrepancy sequences tend to sample space "more uniformly" than random numbers. We use a specific implementation [32, 33] of Sobol sequence that is an adaptation of the INSOBL and GOSOBL routines in ACM TOMS Algorithm 647 [18] and ACM TOMS Algorithm 659 [5]. The routine adapts the ideas of Antonov and Saleev [1]. The original code can only compute the "next" element of the sequence. The revised code allows the user to specify the index of the desired element. The algorithm has a maximum spatial dimension of 40 since MATLAB doesn't support 64 bit integers. A remark by Joe and Kuo shows how to extend the algorithm from the original maximum spatial dimension of 40 up to a maximum spatial dimension of 1111. The FORTRAN90 and C++ versions

of the code has been updated in this way, but updating the MATLAB code has not been simple, since MATLAB doesn't support 64 bit integers. We generate a new quasi-random Sobol vector with each call. The parameters of the algorithm are an integer DIM_{NUM}, the number of spatial dimensions. The algorithm starts with integer $SEED$, the "seed" for the sequence. This is essentially the index in the sequence of the quasi-random value to be generated. On output, $SEED$ has been set to the appropriate next value, usually simply $SEED + 1$. Output is the real $QUASI(DIM_{NUM})$, the next quasi-random vector.

Adaptive algorithm

One group of algorithms, widely used for numerical calculation of multidimensional integrals, are the adaptive algorithms [14, 28]. Most of the adaptive algorithms use a sequence of increasingly finer subdivisions of the original region, chosen to concentrate integrand evaluations on subregions with difficulties. Two main types of subdivision strategies are in common use: local and global sub- division. The main disadvantage of local subdivision strategy is that it needs a local absolute accuracy requirement which will be met after the achievement of the global accuracy requirement. The main advantage of the local subdivision strategy is that it allows a very simple subregion management. Globally adaptive algorithms usually require more working storage than locally adaptive routines, and accessing the region collection is slower. These algorithms try to minimize the global error as fast as possible, independent of the specified accuracy requirement.

The algorithm shows its advantages when the integrand functions are not smooth but need small number of samples to obtain good accuracy. A Monte Carlo integration embedded in a globally adaptive algorithm is able to provide an unbiased estimate of the integral and also probabilistic error bounds for the estimate. In the mean-time it has higher accuracy and faster convergence than the plain Monte Carlo integration as can be seen from the tables below. The only drawback is the higher computational time.

The Adaptive Monte Carlo algorithm that we developed is based on the ideas and results of the importance separation [13], a method that combines the idea of separation of the domain into uniformly small subdomains with the importance sampling approach. The Adaptive method does not use any a priori information about the smoothness of the integrand, but it uses a posteriori information about the variance. The idea of the method consists of the following (see for instance [10, 14]): the domain of integration G is separated into subdomains with identical volume. The interval [0,1] on every dimension coordinate is partitioned into M subintervals, *i.e.,*

$$G = \sum_j G_j, \ j = 1, M^d.$$

Denote by p_j and I_{G_j} the following expressions:

$$p_j = \int_{G_j} p(x)dx, \ I_{G_j} = \int_{G_j} f(x)p(x)dx.$$

Consider now a random point $\xi_i^{(j)} \in G_j$ with a density function $p(x)/p_j$ and in this case

$$I_{G_j} = \mathrm{E}\left[\frac{p_j}{N}\sum_{i=1}^{N} f(\xi_i^{(j)}))\right] = \mathrm{E}\theta_N.$$

We start with a relatively small number M which is given as input data. For every subdomain the integral I_{G_j} and the variance are evaluated. After that the variance is compared with a preliminary given value ε. If the calculated variance in any region is greater than some constant ε, then this region is divided to new M^d subregions, again by partitioning the segment of the region on every coordinate to M subintervals. The algorithm is described below.

Algorithm

1. **Input data**: *number of points N, constant ε (estimation for the variance), constant δ (stop criterion; estimation for the length of subintervals on every coordinate).*
2. **For** $j = 1, \ M^d$:
   - **2.1.** **Calculate** *the approximation of* I_{Ω_j} *and the variance* D_{Ω_j} *in subdomain* Ω_j *based on N independent realizations of random variable* θ_N;

2.2. **If** $(\mathrm{D}_{\Omega_j} \geq \varepsilon)$ **then**

2.2.1. **Choose** *the axis direction on which the partition will perform*,

2.2.2. **Divide** *the current domain into two* $\left(G_{j_1}, G_{j_2}\right)$ *along the chosen direction*,

2.2.3. **If** *the length of obtained subinterval is less than* δ **then go to** *step 2.2.1* **else** $j = j_1$ $\left(G_{j_1}\right.$ *is the current domain*$\left.\right)$ *and* **go to** *step 2.1*;

2.3. **Else if** $(\mathrm{D}_{\Omega_j} < \varepsilon)$ *but an approximation of* $I_{G_{j_2}}$ *has not been calculated yet*, **then** $j = j_2$ $\left(G_{j_2}\right.$ *is the current domain along the corresponding direction*$\left.\right)$ *and* **go to** *step 2.1*;

2.4. **Else if** $(\mathrm{D}_{\Omega_j} < \varepsilon)$ *but there are subdomains along the other axis directions*, **then go to** *step 2.1*;

2.5. **Else** *Accumulation in the approximation* I_N *of I*.

Computational complexity

Let ξ be a random point with probability density function $p(x)$. Introducing the random variable

$$\theta = f(\xi)$$

with mathematical expectation equal to the value of the integral I_{G_j}, then

$$E\theta = \int_{G_j} f(x)p(x)dx.$$

Let $\xi_1, \xi_2, \ldots, \xi_N$ be independent realizations of the random point ξ with probability density function $p(x)$ and $\theta_1 = f(\xi_1), \ldots, \theta_N = f(\xi_N)$. Then an approximate value of I_{G_j} is

$$\hat{\theta}_N = \frac{1}{N}\sum_{i=1}^{N}\theta_i.$$

One can easily see that the computational complexity of the Crude Monte Carlo is linear, because in this simple case we have to choose N random points in the domain and every such choice is at the cost of $O(1)$ operations. One single evaluation of the function in any of these points is also at the cost of $O(1)$ operations.

In the adaptive Monte Carlo algorithm we are doing the same number of operations. For the simple case when $N = 2$ on the first step we have 4 subdomains and

$$\hat{\theta}_N = \frac{1}{N_1}\sum_{i=1}^{N_1}\theta_i + \frac{1}{N_2}\sum_{i=1}^{N_2}\theta_i + \frac{1}{N_3}\sum_{i=1}^{N_3}\theta_i + \frac{1}{N_4}\sum_{i=1}^{N_4}\theta_i,$$

where $N_1 + N_2 + N_3 + N_4 = N$, so we have the same number of operations as the Crude Monte Carlo to estimate an approximate value of I_{G_j}.

In practice when we divide the domain, we choose only $O(1)$ subdomains where the variance is greater than the parameter ε and this choice is independent of the dimensionality N. One can easily see that on every step adaptiveness is not in all subdomains, but only in $O(1)$ subdomains. In the beginning we choose $\frac{N}{k_0}$ random points. On the next step, after we divide the domain into 2^N subdomains, we choose only $O(1)$ subdomains where the variance is greater than the parameter ε and it is important that this choice is independent of the dimensionality N. In these subdomains we choose $\frac{N}{k_1}$ points. To summarize, on the j^{th} step of the adaptive algorithm we choose $O(1)$ subdomains and in them we choose $\frac{N}{k_j}$ points. We have that $\sum_{j=0}^{i}\frac{1}{k_j} = 1$. So for the computational complexity we obtain

$$\frac{N}{k_0} + O(1)\frac{N}{k_1} + \ldots + O(1)\frac{N}{k_i} = NO(1)\left(\sum_{j=0}^{i}\frac{1}{k_j}\right) = NO(1) = O(N).$$

Therefore the computational complexity of the adaptive algorithm is linear.

Lattice sets

More detailed information about this method can be found in the work of Wang and Hickernell [34] and Sloan and Kachoyan in [31]. Let G_s denote the unit cube in s-dimensional space, *i.e.*,

$$G_s = [0,1)^s = \{x = (x_1, \ldots, x_s)\} \mid 0 \leq x_j < 1, j = 1, \ldots, s\}. \tag{8}$$

Let $n_1 < n_2 < \ldots$ be a sequence of positive integers, and let P_{n_l} be any set of n_l points in G_s. (Here a set may have multiple copies of the same point.) For any $r = (r_1, \ldots, r_s) \in G_s$ note that $r_1, \ldots, r_s$ is the volume of the box $[0, r)$. Let $N_{n_l}(r)$ denote the number of points in P_{n_l} lying inside the box $[0, r)$. The discrepancy of the set P_{n_l} is defined as the largest difference between the proportion of points in the box and the volume of the box:

$$D(n_l) := \sup_{r \in G_s} \left| \frac{N_{n_l}(r)}{n_l} - r_1 \ldots r_s \right| \tag{9}$$

This notion was introduced by Weyl (1916). If $D(n_l) = o(1)$ as $n_l \to \infty$, then the sequence of sets P_{n_l} , $n_1 < n_2 < \ldots$ is said to be uniformly distributed on G_s with discrepancy $D(n_l)$. The subscript l is often omitted for simplicity. Not only is the discrepancy a geometric method for measuring uniformity of a set, the discrepancy of a set measures its quality for use in numerical quadrature. The error of this approximation is bounded by the Koksma-Hlawka inequality:

$$\left| \int_{G_s} f(x)dx - \frac{1}{n} \sum_{k=1}^{n} f(x_k) \right| \leq D(n).V(f), \tag{10}$$

where $D(n)$ is the discrepancy of the set $P(n) = \{x_1, \ldots, x_n\}$, and $V(f)$ is the bounded variation of f in the sense of Hardy and Krause. If the integrand is smoother and also periodic, then better error bounds may be obtained, in particular for quadrature rules using lattice point sets.

The lattice S is an infinite set of points in R^s with the following three properties:

1. If x and x' belong to S, then $x + x'$ and $x - x'$ also belongs to S.
2. S contains s linearly independent points.
3. There exists a sphere centered at 0 that contains only 0 itself.

A multiple-integration lattice is a lattice that contains $\mathbb{Z}^s$ as a sub-lattice. By a "lattice rule" then, we shall mean a rule of the form

$$I_N(f) = \frac{1}{N} \sum_{j=0}^{N-1} f(x_j),$$

in which $x_0, \ldots, x_{N-1}$ are all the points of a multiple-integration lattice that lie in $[0, 1]^s$. The cubic lattice is

$$\left\{ \left(\frac{j_1}{n}, \ldots, \frac{j_s}{n} : j_i \in \mathbb{Z}, 1 \leqq i \leqq s \right) \right\},$$

where n is a positive integer. The corresponding lattice rule is the "rectangle rule"

$$I_N(f) = \frac{1}{N} \sum_{j_1=0}^{n-1} \cdots \sum_{j_s=0}^{n-1} f\left(\frac{j_1}{n}, \ldots, \frac{j_s}{n} \right),$$

where $N = n^s$. Because N rises very rapidly with s, the rectangle rule suffers in a very obvious way from the "curse of dimensionality." Note that this rule is equivalent, because of the assumed periodicity, to a product-trapezoidal rule.

Let n be an integer, $n \geq 2$ and $a = (a_1, a_2, \ldots a_s)$ be an integer vector modulo n. A set of the form

$$P_n = \left\{ \left\{ \frac{ak}{n} \right\} = \left(\left\{ \frac{a_1 k}{n} \right\}, \ldots, \left\{ \frac{a_s k}{n} \right\} \right) \mid k = 1, \ldots, n \right\}$$

is called a lattice point set, where $\{x\}$ denotes the fractional part of x. The vector a is called called a lattice point or generator of the set. As one can see, the formula for the lattice point set is simple to program. The difficulty lies in finding a good value of a, such that the points in the set are evenly spread over the unit cube. The choice of good

generating vector, which leads to small errors, is not trivial. Complicated methods from theory of numbers are widely used, for example Zaremba's index or error of the worst function. Korabov consider the following vectors:

$$a = (1, a, a^2, \ldots, a^{s-1}) \bmod N, 1 \leq a \leq N-1, gcd(a, N) = 1.$$

The method can be applied only for for number of points $n_l = F_l^{(s)}$, *i.e.,* only for generalized Fibonacci number of points. This set used the generating vector

$$a = (1, F_{l+1}^{(s)}, ..., F_{l+s-1}^{(s)})), \quad n_l = F_l^{(s)},$$

where $F^{(s)}$ is the corresponding generalized Fibonacci number of dimensionality s:

$$F_{l+s}^{(s)} = F_l^{(s)} + F_{l+1}^{(s)} + ... + F_{l+s-1}^{(s)}, l = 0, 1, \ldots$$

with initial conditions:

$$F_0^{(s)} = F_1^{(s)} = \ldots = F_{s-2}^{(s)} = 0, F_{s-1}^{(s)} = 1,$$

for $l = 0, 1, \ldots$.

The discrepancy of the set obtained by using the vector described above is asymptotically estimated in [34]. We have the following estimation:

$$D(n_l) = O(n_l^{-\frac{1}{2} - \frac{1}{2^{s+1}.\log 2} - \frac{1}{2^{2s+3}}}).$$

In [24] is mentioned that Fibonacci lattice sets are very appropriate for smoothness functions. The advantage of the lattice method is the linear computational complexity and reduced time for calculating the multidimensional integrals. The number of calculation required to obtain the generating vector is asymptotically less than $O(N_l)$. The generation of a new point requires constant number of operations thus to obtain a lattice set of the described kind consisting of N_l points, $O(N_l)$ number of operations are necessary.

NUMERICAL EXAMPLES AND RESULTS

Our experimental results include the evaluation of the following 3,5,8 and 20 dimensional integrals:

$$\int_{[0,1]^3} \exp(x_1 x_2 x_3) \approx 1.14649907. \tag{11}$$

$$\int_{[0,1]^5} \exp(\sum_{i=1}^{5} 0.5 a_i x_i^2 (2 + \sin \sum_{j=1, j\neq i}^{5} x_j)) \approx 2.923651, a_i = (1, 0.5, 0.2, 0.2, 0.2). \tag{12}$$

$$\int_{[0,1]^8} \exp(\sum_{i=1}^{8} 0.1 x_i) = 1.496805. \tag{13}$$

$$\int_{[0,1]^{20}} \exp(\prod_{i=1}^{20} x_i) \approx 1.00000949634. \tag{14}$$

The results are given in the tables below. Each table contains information about the MC approach which is applied, the obtained relative error, the needed CP-time and the number of points. Note that when the lattice method is tested, all of these numbers are Generalized Fibonacci numbers of the corresponding dimensionality. We have used CPU Intel Core i5-2410M @ 2.30GHz and MATLAB. The advantage of the Lattice method for low dimensions is superior. For higher dimensions Sobol sequence has advantages for fixed number of points and it is slightly worse than Lattice for a fixed computational time. The Adaptive Monte Carlo is better than Crude Monte Carlo after some seconds, and it strength is when the integrand have some pecularities like the Genz test functions [20].

The advantage of the lattice method for the (11) integral is superior, it gives $1.32e-6$ for 0.1s and $3.22e-7$ for 1s in Table 2, while Sobol needs 20s to achieve the same accuracy. While for more computational time Sobol improves

Table 1. The relative error for 3 dimensional integral

N	Crude	time	adaptive	time	lattice	time	Sobol	time
19513	8.93e-4	0.01	3.21e-4	2.21	4.69e-4	0.02	4.98e-5	0.56
35890	2.18e-3	0.04	6.55e-5	6.41	5.46e-6	0.06	1.56e-5	1.45
66012	5.65e-4	0.07	5.12e-5	9.86	5.34e-6	0.11	8.11e-6	2.31
121415	6.46e-4	0.12	5.11e-5	15.4	5.34e-6	0.12	3.08e-6	3.80
223317	4.15e-4	0.20	9.34e-5	24.2	1.73e-6	0.22	2.05e-6	6.13

Table 2. Times for 3 dimensional integral

time,s	crude	adaptive	lattice	sobol
0.1	6.56e-4	8.67e-4	1.32e-6	3.21e-4
1	1.37e-4	2.96e-5	3.22e-7	8.21e-5
2	5.29e-5	5.45e-4	2.06e-7	2.96e-5
5	1.84e-4	1.14e-4	1.47e-7	5.00e-6
10	7.79e-5	6.56e-5	3.89e-7	2.71e-6
20	4.57e-5	2.04e-5	1.53e-8	1.88e-6

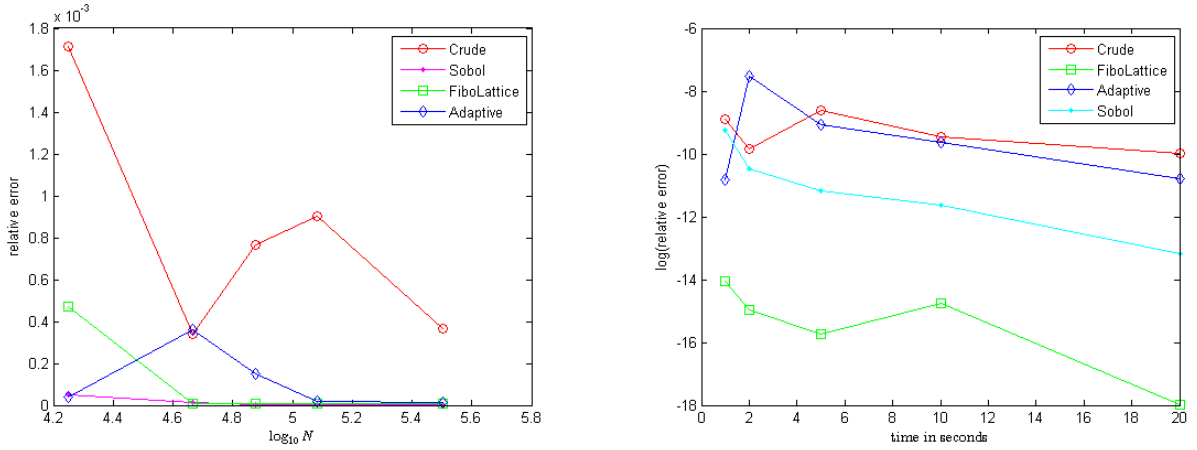

Figure 1. Relative error and computational time for 3-dimensional integral with Monte Carlo and quasi Monte Carlo methods

Table 3. The relative error for 5 dimensional integral

N	Crude	time	adaptive	time	lattice	time	Sobol	time
13624	6.72e-3	0.02	1.89e-3	2.33	9.59e-4	0.03	1.76e-4	0.56
52656	2.53e-3	0.06	2.31e-3	6.18	6.96e-4	0.06	5.05e-5	1.45
103519	2.48e-3	0.09	2.01e-3	9.94	8.72e-5	0.13	2.70e-5	2.52
203513	2.15e-3	0.15	3.42e-4	16.2	8.04e-5	0.25	7.57e-6	6.07
400096	1.66e-3	0.40	9.12e-4	45.6	7.26e-5	0.50	2.52e-6	10.63

a lot, Adaptive and Crude algorithms performs in similar way and the results are worse and as can be seen Adaptive Monte Carlo is better than Crude Monte Carlo after 5s.

Applying the methods to the (12) integral in Table 4 Lattice method gives best results- $3.85e-7$ for 20s, and the Sobol sequence gives $4.96e-6$ for the same time. However for a fixed number of points in Table 3 Sobol algorithm is more accurate. Adaptive and Crude are closer and gives worse results.

In Table 6 for the (13) integral Lattice method has advantage over the Sobol and Adaptive, for 1s it produces error $5.34e-6$ which is the same result as Sobol for 10s. Here Adaptive algorithm performs in similar way to Crude algorithm.

Table 4. Times for 5 dimensional integral

time,s	crude	adaptive	lattice	sobol
0.1	3.07e-3	1.34e-2	7.26e-5	8.22e-4
1	1.32e-3	2.44e-3	2.28e-5	2.91e-4
5	1.43e-3	4.93e-4	5.94e-6	1.71e-5
10	8.47e-5	1.88e-3	3.85e-7	1.79e-5
20	2.52e-4	2.71e-4	7.49e-7	4.96e-6

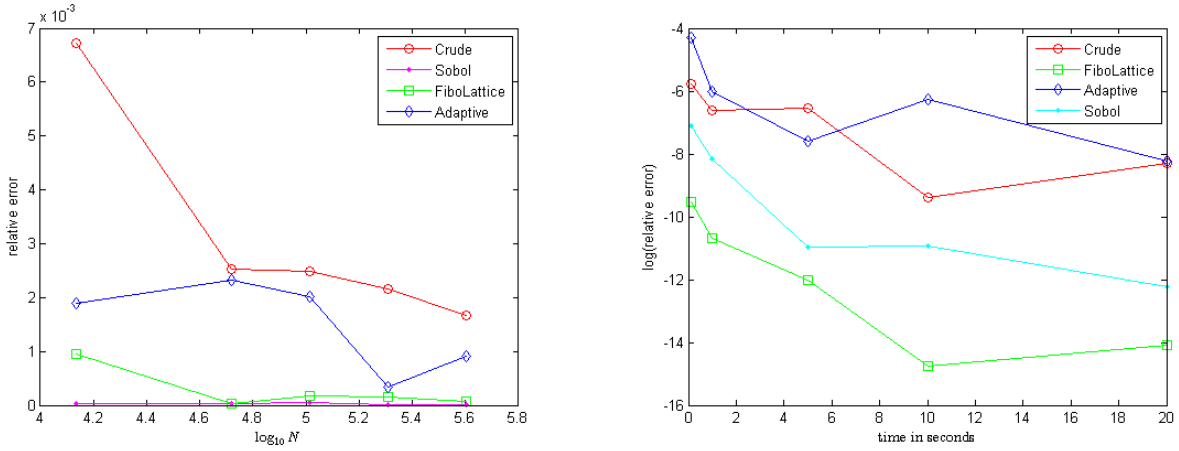

Figure 2. Relative error and computational time for 5-dimensional integral with Monte Carlo and quasi Monte Carlo methods

Table 5. The relative error for 8 dimensional integral

N	Crude	time	adaptive	time	lattice	time	Sobol	time
16128	1.79e-3	0.04	1.10e-5	12.6	8.08e-4	0.03	8.87e-5	0.13
32192	8.76e-4	0.05	3.32e-5	33.3	1.03e-4	0.07	5.42e-5	0.58
64256	2.00e-4	0.08	4.65e-5	54.2	5.03e-5	0.11	2.34e-5	2.49
128257	2.33e-4	0.13	8.25e-6	88.3	8.13e-6	0.14	4.45e-6	6.36
510994	2.21e-4	0.34	7.07e-6	233.6	5.95e-6	0.57	3.32e-6	19.45

Table 6. Times for the 8 dimensional integral

time,s	crude	adaptive	lattice	sobol
1	2.68e-4	6.34e-4	5.34e-6	2.02e-5
2	5.32e-5	1.58e-4	2.57e-6	2.73e-5
5	1.03e-5	1.44e-4	1.52e-7	8.88e-6
10	5.17e-5	6.61e-5	3.45e-6	5.23e-6
20	1.11e-5	2.77e-5	1.82e-7	2.11e-6

We expect that for larger dimension Sobol is the best but again Lattice method has minor advantage over the Sobol for a fixed time for the (14) integral. In Table 8, for 1s it produces error $1.48e-5$, which is the same result as Sobol for 5s. However, for the same number of points Sobol gives better accuracy. Here Adaptive algorithm performs better than Crude algorithm and has better accuracy for the same number of points closer to Lattice method.

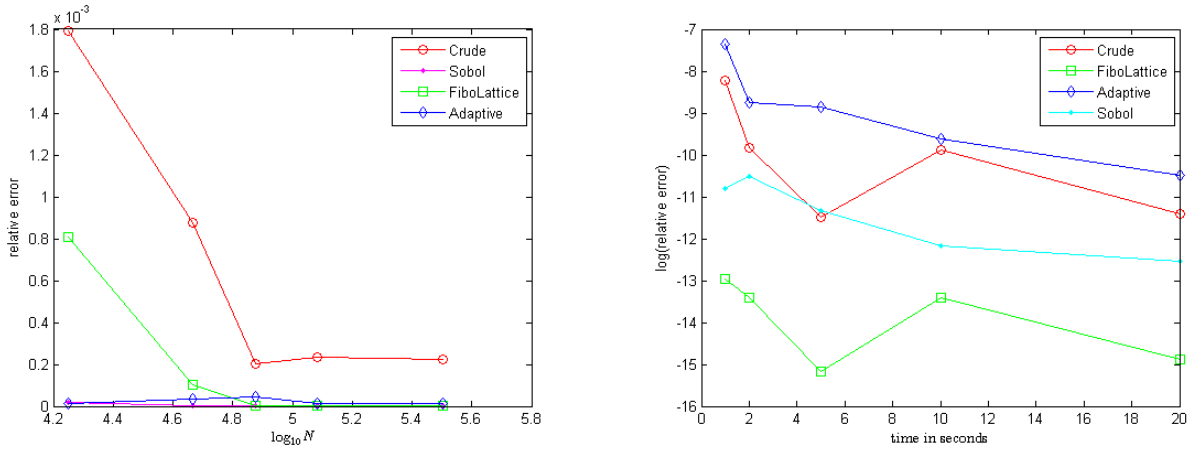

Figure 3. The relative error and computational time for 8-dimensional integral with Monte Carlo and quasi Monte Carlo methods

Table 7. The relative error for 20 dimensional integral

N	Crude	time	adaptive	time	lattice	time	Sobol	time
2048	2.84e-2	0.02	1.14e-2	8.6	8.22e-5	0.03	8.44e-4	0.13
16384	8.23e-4	0.12	4.96e-4	60.3	3.12e-5	0.13	6.82e-5	1.68
65536	8.61e-3	0.91	9.75e-4	474.2	1.36e-5	1.17	8.34e-6	8.69
131072	4.13e-4	2.13	1.25e-5	888.3	8.85e-6	2.34	3.77e-6	14.36
524288	1.22e-4	8.13	1.96e-6	2356	2.15e-6	8.34	1.91e-7	57

Table 8. Times for the 20 dimensional integral

time,s	crude	adaptive	lattice	sobol
1	9.14e-3	1.58e-3	1.48e-5	3.25e-5
2	3.68e-3	1.028e-3	9.17e-6	3.97e-5
5	2.67e-3	8.58e-4	5.19e-6	1.45e-5
10	3.34e-4	4.02e-4	1.73e-6	2.71e-6
20	1.53e-4	1.13e-4	1.38e-7	1.76e-6

CONCLUSION

Four completely different approaches are applied in finance for European option pricing. The value of the option is formulated in terms of the expectation of some random variable and the average of independent samples of this random variable is used to estimate the value of the option. This paper shows that a particular type of low discrepancy sequence known as lattice rules have strong advantages in the case of low to medium dimension problems as long as the integrands are sufficiently regular. The quasi-random Sobol sequence is slower but gives closer accuracy, being better for a fixed number of points with increasing the dimensionality. The Adaptive algorithm requires more computational time to achieve better accuracy, gives good results regardless of the dimensions and it is the best method when the integrand functions are not smooth. The plain(Crude) Monte Carlo algorithm using the Mersenne Twister number generator gives worst, but fastest results and it is a choice when it is sufficient an error of 1%. The four approaches are completely different thus it is a question of interest to know which one of them outperforms the other. The main advantage of the presented algorithms is their linear computational complexity regardless of the dimension, where the deterministic methods suffer from the so-called curse of dimensionality and become impractical.

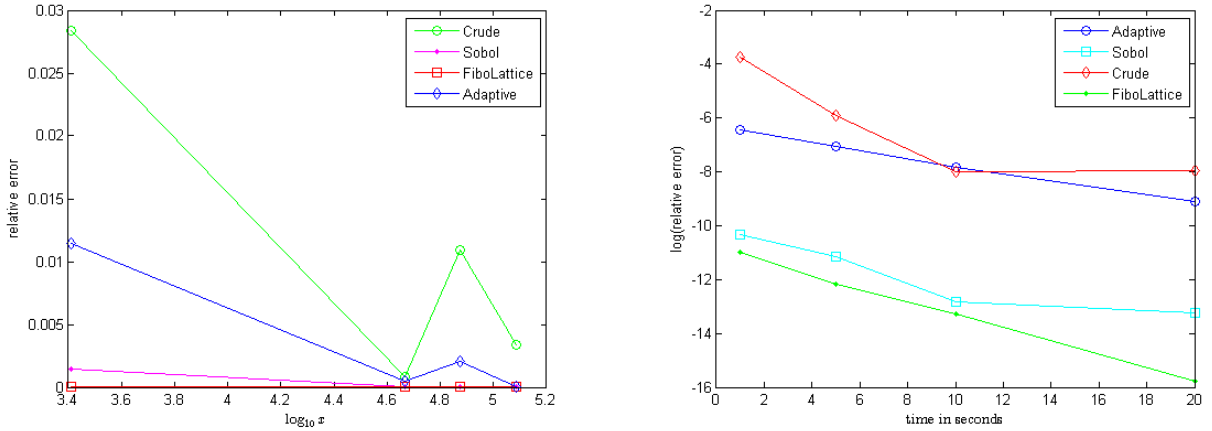

Figure 4. Relative error and computational time for 20-dimensional integral with Monte Carlo and quasi Monte Carlo methods

ACKNOWLEDGEMENTS

This work was supported by the Program for career development of young scientists, BAS Grant No. DFNP-91/04.05.2016, and partially by the Bulgarian National Science Fund under grant DFNI I02-20/2014.

References

[1] I. A. Antonov and V. M. Saleev (1980) An Economic Method of Computing LP Tau-Sequences, *USSR Computational Mathematics and Mathematical Physics* **19**, 252–256.
[2] F. Black and M. Scholes (1973) *J. Pol. and Econ.* **81**, 637–659.
[3] P. P. Boyle (1977) *J. Finan. Econ.* **4**, 323–338.
[4] P. Boyle, Y. Lai, and K. Tan, *Using Lattice Rules to Value Low-Dimensional Derrivative Contracts*, 2001.
[5] P. Bratley and B. Fox (1988) *ACM Transactions on Mathematical Software* **14**(1), 88–100.
[6] M. Broadie and P. Glasserman (1997) *J. of Economic Dynamics and Control* **21**, 1323–1352.
[7] D. M. Chance, *An Introduction to Derivatives* 3rd edn (The Dryden Press, 1995).
[8] J. C. Cox, M. Rubinstein *Options Markets* (Prentice Hall, 1985).
[9] J. C. Cox, S. A. Ross, and M. Rubinstein (1979) *J. Fin. Econ.* **7**, 229–263.
[10] I. Dimov *Monte Carlo Methods for Applied Scientists* (New Jersey, London, Singapore, World Scientific, 2008, 291p.).
[11] I. Dimov and E. Atanassov (2007) *Applied Mathematical Modelling* **32**, 1477–1500.
[12] I. Dimov, R. Georgieva, and V. Todorov, "Balancing of systematic and stochastic errors in Monte Carlo algorithms for integral equations," in *Proc. 8th Int. Conf. Num. Methods and Appl., NMA 2014, Borovets, Bulgaria, August 20-24, 2014*, *LNCS***89622**, pp.44–51.
[13] I. Dimov and A. Karaivanova (1998) *Mathematics and Computers in Simulation* **47**, 201–213.
[14] I. Dimov, A. Karaivanova, R. Georgieva, and S. Ivanovska, "Parallel importance separation and adaptive Monte Carlo algorithms for multiple integrals," in *Proc. 5th Int. Conf. Num. Methods and Appl., August, 2002, Borovets, Bulgaria*, *LNCS***2542**, (Springer-Verlag, Berlin, Heidelberg, New York, 2003), pp. 99–107.
[15] D. Duffie *Dynamic Asset Pricing Theory* (Princeton, 1992).
[16] D. Duffie *Security Markets: Stochastic Models* (Academic Press, Inc., 1988).
[17] R. Eckhardt, *Stan Ulam John von Neumann and the Monte Carlo Method*. 1987.
[18] B. Fox, (1986) *ACM Transactions on Mathematical Software* **12**(4), 362–376.
[19] G. Gemmill *Options Pricing* (McGraw-Hill, 1992).
[20] A. Genz, *Testing Multidimensional Integration Routines. Tools Methods and Languages for Scientific and Engineering Computation*, 1984, pp.81–94.
[21] J. M. Harrison and S. R. Pliska (1981) *Stoch. Proc. Appl.* **11**, 261–271.

[22] L. K. Hua and Y. Wang *Applications of Number Theory to Numerical analysis*, 1981.
[23] J. C. Hull *Options, Futures, and other Derivative Securities* (Prentice-Hall, Inc. 1993).
[24] A. Karaivanova, *Stochastic Numerical Methods and Simulations*, 2012.
[25] Y. Lai, "Monte Carlo and Quasi-Monte Carlo Methods and Their Applications," Ph.D. dissertation, Claremont Graduate University, 1998.
[26] Y. Lai, J. Spanier, "Applications of Monte Carlo/Quasi-Monte Carlo Methods in finance: Option Pricing," in *Proceedings of a Conference held at the Claremont Graduate Univ., 1998.*
[27] H. Niederreiter and D. Talay, *Monte Carlo and Quasi-Monte Carlo Methods*, 2010.
[28] K. Osterlee and X. Huang (2009) *J. Comp. Appl. Math.* **231**, 506–516.
[29] S. Paskov, "Computing high dimensional integrals with applications to finance," preprint Columbia Univ., 1994.
[30] I. H. Sloan and S. Joe, *Lattice Methods for Multiple Integration* (Oxford University Press, Oxford, 1994).
[31] I. H. Sloan and P. J. Kachoyan (1987) *SIAM Journal of Numeracal Analysis* **14**, 116–128.
[32] I. Sobol (1977) "USSR Computational Mathematics and Mathematical Physics," **16**, 236–242.
[33] I. Sobol and Levitan, "The Production of Points Uniformly Distributed in a Multidimensional Cube," Preprint IPM Akad. Nauk SSSR, Number 40, Moscow 1976. [in Russian]
[34] Y. Wang and F. J. Hickernell, *An Historical Overview of Lattice Point Sets*, 2002
[35] P. Wilmott, J. Dewynne, and S. Howison, *Option Pricing: Mathematical Models and Computation* (Oxford University Press, 1995).