Monte Carlo Algorithms for Linear Problems

Ivan Dimov

Central Laboratory for Parallel Computing
Bulgarian Academy of Sciences
Acad. G. Bonchev St.,bl. 25 A, Sofia, 1113, Bulgaria
E-mail: dimov@copern.acad.bg
Web site: http://www.acad.bg/BulRTD/math/dimov2.html

Key words: Monte Carlo algorithms, linear problems, boundary value problem, efficiency estimator, Markov chain, parallel algorithms

MSC subject classification: 65 C 05, 65 U 05

C O N T E N T S

- Introduction
- Iterative Monte Carlo Algorithms
- Solving Linear Systems and Matrix Inversion.
- Monte Carlo Algorithms for Boundary Value Problems
- Applications: Sensitivity Analysis in Environmental Mathematics

1 Introduction

Monte Carlo methods are a powerful tool in many fields of mathematics, physics and engineering. It is known, that these methods give statistical estimates for the functional of the solution by performing random sampling of a certain chance variable whose mathematical expectation is the desired functional.

Monte Carlo methods are methods for solving problems using random variables. In the book [Sh66] edited by Yu.A.Shreider one can find the following definition of the Monte Carlo method.

Definition 1.1 *The Monte Carlo method consists of solving various problems of computational mathematics by means of the construction of some random process for each such problem, with the parameters of the process equal to the required quantities of the problem.*

Usually Monte Carlo methods reduce problems to the approximate calculation of their mathematical expectations.

The year 1949 in generally regarded as the official birthday of the Monte Carlo methods when the paper of Metropolis and Ulam [MU49] was published, although some authors point to earlier dates. Ermakov [Er84], for example, notes that a solution of a problem by the Monte Carlo method is contained in the Old Testament. The development of the method is connected with the names of John von Neumann, E. Fermi and G. Kahn, who worked at Los Alamos (USA) for forty years. The development of modern computers and particularly parallel computing systems, provided fast and specialized generators of random numbers and gave a new momentum to the development of Monte Carlo methods.

There are two main directions in development and studying Monte Carlo algorithms. The first direction is *Monte Carlo simulation* algorithms. These algorithms are used for *simulation* of real-life processes and phenomena. In this case, the algorithm just follows the corresponding physical, chemical or biological processes which are have to be studied. In such simulations Monte Carlo is used as a tool for choosing one of different possibilities of going of the processes. Monte Carlo simulation could be considered as a method for solving *probabilistic* problems using some kind of simulations of *random variables* or *random fields*.

The second direction is *Monte Carlo numerical* algorithms. The Monte Carlo numerical algorithms are usually used for solving *deterministic* problem by modeling random variables or random fields. The main idea here is to construct some *artificial* random process and to prove that the mathematical expectation of the process is equal to the unknown solution of the problem or to the some functional of the solution. Usually, there are more the one possibility to create such an artificial process. After finding such a process one needs to define an algorithm for computing realizations of the random variable. Usually, the random variable can be considered as a weight of a random process (usually, a Markov process). Then, The Monte Carlo algorithm for solving the problem under consideration consist in simulation the Markov process and computing the realizations of the random variables.

Here the following important problems arise:

- How to define the random process for which the mathematical expectation is equal to the unknown solution?
- How to estimate the statistical error?
- How to choose the random process such that to have *high computational efficiency* solving the problem with a given statistical accuracy (for an apriori given probability error)?

In this paper algorithms of the second direction, i.e. *Monte Carlo numerical* algorithms will be considered. The general scheme of our consideration is the following

- We define the problem under consideration and show the conditions which have to be satisfied for obtaining the unique solution.

- Construct a random process and prove that such a process can be used for obtaining the approximate solution of the problem.
- Estimate the probability error of the method.
- Try to find the optimal (in some sense) algorithm, that is to choose the random process for which the statistical error is minimal.
- Choose the parameters of the algorithm (such that the number of the realizations of the random variable, the length (number of states) of the Markov chain and, so on) in order to have a good balance between the *statistical* and the *systematic* errors.
- Obtain a priori estimates for the *speed-up* and the *parallel efficiency* of the algorithm when *parallel or vector machines* are used.

An important advantage of the Monte Carlo algorithms is that they allow to find directly the unknown functional of the problem solution with a number of operations, necessary to solve the problem in one point of the domain [So73], [DT93]. Often, one do not need to know the solution in the whole domain in which the problem is defined. Usually, it is necessary to know the value of some functional of the solution. Problems of this kind could be found in many field of the applied sciences. For example, in the Statistical Physics, one is interested in computing linear functional of the solution of the equations for density distribution function (such as Boltzmann , Wigner or Schroedinger equation), i.e., probability to find the particle in a given place of the space and in a given time (integral of the solution), mean value of the velocity of the particles (the first integral moment of the velocity) or the energy (the second integral moment of the velocity) and, so on.

It is well known that, Monte Carlo methods are very efficient when parallel processors or parallel computers are available. This is because these methods are inherently parallel and have loose dependencies. In addition, they are also well vectorizable when powerful vector machines are used. Nevertheless, the problem of parallelization of the Monte Carlo methods is not a trivial problem because different kinds of parallelization can be used. To find the most efficient parallelization in order to obtain a high value of the speed-up of the algorithm is very important practical problem.

2 Iterative Monte Carlo algorithms

In general, Monte Carlo numerical algorithms may be divided into two classes – *direct* algorithms and *iterative* algorithms. The direct algorithms provide an estimate of the solution of the equation in a finite number of steps, and contain only a stochastic error. For example, direct Monte Carlo algorithms are the algorithms for evaluating integrals [HH64], [So73]. Iterative Monte Carlo algorithms deal with an approximate solution obtaining an improved solution with each step of the algorithm. In principle, they require an infinite number of steps to obtain the exact solution, but usually one is happy with an approximation to say t significant figures. In this latter case there are two errors - *stochastic* and *systematic*. The systematic error depends both on the number of iterations performed and the characteristic values of the iteration matrix, while the stochastic errors depend on the probabilistic nature of the algorithm.

Iterative algorithms are preferred for solving integral equations and large sparse systems of algebraic equations (such as those arising from approximations of partial differential equations). Such algorithms are good for diagonally dominant systems in which convergence is rapid; they are not so useful for problems giving use to dense matrices, for example.

Define an iteration of degree j as

$$u^{(k+1)} = F_k(A, b, u^{(k)}, u^{(k-1)}, \ldots, u^{(k-j+1)}),$$

where $u^{(k)}$ is obtained from the kth iteration. It is desired that

$$u^{(k)} \rightarrow u = A^{-1}b \quad \text{as} \quad k \rightarrow \infty.$$

Usually the degree of j is kept small because of storage requirements.

The iteration is called *stationary* if $F_k = F$ for all k, that is, F_k is independent of k.

The iterative Monte Carlo process is said to be *linear* if F_k is a linear function of $u^k, \ldots, u^{(k-j+1)}$.

We shall consider *iterative stationary linear Monte Carlo algorithms* and will analyse both systematic and stochastic errors.

Consider a general description of the iterative Monte Carlo algorithms. Let $\mathbf{X}$ be a Banach space of real-valued functions. Let $f = f(x) \in \mathbf{X}$ and $u_k = u(x_k) \in \mathbf{X}$ be defined in $I\!R^d$ and $L = L(u)$ be a linear operator defined on $\mathbf{X}$.

Consider the sequence $u_1, u_2, \ldots$, defined by the recursion formula

$$u_k = L(u_{k-1}) + f, \quad k = 1, 2, \ldots \tag{1}$$

The formal solution of (1) is the truncated Neumann series

$$u_k = f + L(f) + \ldots + L^{k-1}(f) + L^k(u_0), \qquad k > 0, \tag{2}$$

where L^k means the k-th iterate of L.

As an example consider the integral iterations.

Let $u(x) \in \mathbf{X}$, $x \in G \subset I\!R^d$ and $l(x, x')$ be a function defined for $x \in G, x' \in G$. The integral transformation

$$Lu(x) = \int_G l(x, x')u(x')dx'$$

maps the function $u(x)$ into the function $Lu(x)$, and is called an *iteration of* $u(x)$ *by the integral transformation kernel* $l(x, x')$. The second integral iteration of $u(x)$ is denoted by $LLu(x) = L^2u(x)$.

Obviously, $L^2u(x) = \int_G \int_G l(x, x')l(x', x'')dx'dx''$.

In this way $L^3u(x), \ldots, L^iu(x), \ldots$ can be defined.

When the infinite series converges, the sum is an element u from the space $\mathbf{X}$ which satisfies the equation

$$u = L(u) + f. \tag{3}$$

The truncation error of (2) is $u_k - u = L^k(u_0 - u)$. Let $J(u_k)$ be a linear functional that is to be calculated. Consider the spaces

$$\mathbf{T}_{i+1} = \underbrace{I\!R^d \times I\!R^d \times \ldots \times I\!R^d}_{i \text{ times}}, \qquad i = 1, 2, \ldots, k, \tag{4}$$

where "$\times$" denotes the Cartesian product of spaces.

Random variables $\theta_i,\ i = 0, 1, \ldots, k$ are defined on the respective product spaces $\mathbf{T_{i+1}}$ and have conditional mathematical expectations:

$$E\theta_0 = J(u_0),\ E(\theta_1/\theta_0) = J(u_1), \ldots, E(\theta_k/\theta_0) = J(u_k),$$

where $J(u)$ is a linear functional of u.

The computational problem then becomes one of calculating repeated realizations of θ_k and combining them into an appropriate statistical estimator of $J(u_k)$.

As an approximate value of the linear functional $J(u_k)$ is set up

$$J(u_k) \approx \frac{1}{n} \sum_{s=1}^{n} \{\theta_k\}_s, \tag{5}$$

where $\{\theta_k\}_s$ is the s-th realization of the random variable θ_k.

The probable error r_n of (5) [EM82] is then

$$r_n = c\ s(\theta_k) n^{-\frac{1}{2}},$$

where $c \approx 0.6745$ and $s(\theta_k)$ is the standard deviation of the random variable θ_k.

There are two approaches which correspond to two special cases of the operator L :

(i) L is a matrix and u and f are vectors;

(ii) L is an ordinary integral transform

$$L(u) = \int_G l(x, y) u(y) dy.$$

First consider the second case. Equation (3) becomes

$$u(x) = \int_G l(x, y) u(y) dy + f(x) \ \text{ or } \ u = Lu + f. \tag{6}$$

Monte Carlo algorithms frequently involve the evaluation of linear functionals of the solution of the following type

$$J(u) = \int_G h(x) u(x) dx = (u, h). \tag{7}$$

In fact, the equation (7) defines an inner product of a given function $h(x) \in \mathbf{X}$ with the solution of the integral equation (3).

Sometimes, the adjoint equation

$$v = L^* v + h \tag{8}$$

will be used.

In (8) $v, h \in \mathbf{X}^*$, $L^* \in [\mathbf{X}^* \to \mathbf{X}^*], \mathbf{X}^*$ is the adjoint functional space to $\mathbf{X}$ and L^* is an adjoint operator.

For some important applications $\mathbf{X} = \mathbf{L_1}$ and

$$\| f \|_{\mathbf{L}_1} = \int_G | f(x) | \, dx; \quad \| L \|_{\mathbf{L}_1} \leq \sup_x \int_G | l(x, x') | \, dx' \tag{9}$$

In this case $h(x) \in \mathbf{L}_\infty$, hence $\mathbf{L}_1{}^* \equiv \mathbf{L}_\infty$ and $\| h \|_{\mathbf{L}_\infty} = \sup | h(x) \ \ x \in G$.

For many applications $\mathbf{X} = \mathbf{X}^* = \mathbf{L}_2$. Note also, that if $h(x), u(x) \in \mathbf{L}_2$ then the inner product (7) is finite. In fact,

$$\left| \int_G h(x)u(x)dx \right| \leq \int_G |h(x)u(x)|dx \leq \left\{ \int_G h^2 dx \int_G u^2 dx \right\}^{1/2} < \infty$$

One can see, that if $u(x) \in \mathbf{L}_2$ and $l(x, x') \in \mathbf{L}_2(G \times G)$ then $Lu(x) \in \mathbf{L}_2$:

$$|Lu(x)|^2 \leq \left\{ \int_G |lu| dx' \right\}^2 \leq \int_G l^2(x, x')dx' \int_G u^2(x')dx'.$$

Let us integrate the last inequality with respect to x:

$$\int_G |Lu|^2 dx \leq \int_G \int_G l^2(x, x')dx'dx \int_G u^2(x')dx' < \infty.$$

From the last inequality it follows that $L^2u(x), \ldots, L^iu(x), \ldots$ belong also to $\mathbf{L}_2(G)$.

Obviously, if $u \in \mathbf{L}_1$ and $h \in \mathbf{L}_\infty$ the inner product (7) will be bounded.

If it is assumed that $\| L^m \| < 1$, where m is any natural number, then the Neumann series $u = \sum_{i=0}^{\infty} L^i f$ converges.

The condition $\| L^m \| < 1$ is not very strong, since, as it will be shown in this section, it is possible to construct a Monte Carlo method for which the Neumann series does not converge. Analytically extending the resolvent by a change of the spectral parameter gives a possibility to obtain a convergent method when Neumann series for the original problem does not converge or to accelerate the rate of convergence when it converges slowly.

Consider the Monte Carlo method for evaluation the functional (7). It can be seen that when $l(x, x') \equiv 0$ evaluation of the integrals can pose a problem. Consider a random point $\xi \in G$ with a density $p(x)$ and let there be n realizations of the random point $\xi_i (i = 1, 2, ..., n)$. Let a random variable $\theta(\xi)$ be defined in G, such that $E\theta(\xi) = J$.

Then the computational problem becomes one of calculating repeated realizations of θ and of combining them into an appropriate statistical estimator of J. Note that the nature of the every process realization of θ is a Markov process. We will consider only *discrete Markov processes with a finite set of states*, the so called *Markov chains* $T_i\{k_1, k_2, ..., k_i\}$. At each of a discrete sequence of times $t = 0, 1, \ldots, i$ the system T_i is in one of the states k_j. The state k_j determines a set of conditional probabilities p_{jl}, such that p_{jl} is the probability that the system, which at the τth time, is in the state k_j, will be in the state k_l at the $\tau + 1$th time.

Thus, p_{jl} is the probability of the transition $k_j \Rightarrow k_l$. The set of all conditional probabilities p_{jl} define a matrix $P = \{p_{jl}\}_{j,l=1}^{i}$ which completely determines the probabilities of the given chain T_i.

Definition 2.1 *The state is called absorbing if the chain terminates in this state with probability* 1.

In the general case, iterative Monte Carlo algorithms can be defined as *terminated Markov chains*:

$$T = k_{i_0} \rightarrow k_{i_1} \rightarrow k_{i_2} \rightarrow \ldots k_{i_t}, \tag{10}$$

where k_{i_t} is one of the absorbing states. This determines the value of some function $F(T)$, which depends upon the sequence (10). The function $F(T)$ is a random variable. After the value of $F(T)$ has been calculated, the system is restarted to its initial state k_{i_0} and the transitions are begun anew. A number of n independent runs are made through the Markov chain starting from the state k_{i_0} to any of the absorbing states. The average

$$\frac{1}{n}\sum_{T} F(T) \tag{11}$$

is taken over all actual sequences of transitions (10). The value in (11) approximates $E\{F(T)\}$, which is the required functional of the solution.

We also will be interested in *algorithmic complexity*.

Definition 2.2 *The algorithmic complexity is define as a value*

$$n \times E(\tau)t_0,$$

where $E(\tau)$ is the mathematical expectation of the number of transitions in the sequence (10) and t_0 is the mean time needed for realization of one transition.

An approximate value of the linear functional J, defined by (7) is

$$J \approx \frac{1}{n}\sum_{s=1}^{n}(\theta)_s = \hat{\theta}_n,$$

where $(\theta)_s$ is the s-th realization of the random variable θ.

The random variable whose mathematical expectation is equal to $J(u)$ is given by the following expression

$$\theta[h] = \frac{h(\xi_0)}{p(\xi_0)} \sum_{j=0}^{\infty} Q_j f(\xi_j),$$

where $Q_0 = 1$; $Q_j = Q_{j-1}\dfrac{l(\xi_{j-1}, \xi_j)}{p(\xi_{j-1}, \xi_j)}$, $j = 1, 2, \ldots,$ and $\xi_0, \xi_1, \ldots$ is a Markov chain in G with initial density function $p(x)$ and transition density function $p(x, y)$.

For the first case, when the linear operator L is a matrix, the equation (2) can be written in the following form :

$$u_k = L^k u_0 + L^{k-1} f + \ldots + Lf + f = (I - L^k)(I - L)^{-1} f + L^k u_0, \tag{12}$$

where I is the unit (identity) matrix;

$L = (l_{ij})_{i,j=1}^{m}$; $u_0 = (u_1^0, \ldots, u_m^0)$ and matrix $I - L$ is supposed to be non-singular.

It is well known that if all eigenvalues of the matrix L lie within the unit circle of the complex plane there exists a vector u such that $u = \lim_{k\to\infty} u_k$, which satisfies the equation

$$u = Lu + f \tag{13}$$

(see, for example, [GV83]).

Now consider the problem of evaluating the inner product

$$J(u) = (h, u) = \sum_{i=1}^{m} h_i u_i \,, \tag{14}$$

where $h \in I\!R^m$ is a given vector .

To construct a random variable whose mathematical expectation coincides with functional (14) for the system (16 first consider the integral equation (6) for which $G = [0, m)$ is an one-dimensional interval divided into equal subintervals $G_i = [i-1, i)$, $i = 1, 2, \ldots m$ such that

$$\begin{cases} l(x, y) = l_{ij} & , \ x \in G_i, \ y \in G_j \\ f(x) = f_i & , \ x \in G_i \end{cases}$$

Then the integral equation (6) becomes $u_i = \sum_j \int_{G_j} l_{ij} u(y) dy + f_i$ for $u_i \in G_i$. Denote

$$u_j = \int_{G_j} u(y) dy \tag{15}$$

so that one obtains, for $u(x) \in G_i$, $u(x) = \sum_{j=1}^{m} l_{ij} u_j + f_i$. From the last equation it follows that $u(x) = u_i$ and so, $u_i = \sum_{j=1}^{m} l_{ij} u_j + f_i$, or in a matrix form

$$u = Lu + f \,, \tag{16}$$

where $L = \{l_{ij}\}_{i,j=1}^{m}$.

The above permits the construction of the following random variable

$$\theta[h] = \frac{h_{k_0}}{p_0} \sum_{\nu=0}^{\infty} Q_\nu f_{k_\nu} \,, \tag{17}$$

where

$$Q_0 = 1; \qquad Q_\nu = Q_{\nu-1} \frac{l_{k_{\nu-1},k_\nu}}{p_{k_{\nu-1},k_\nu}}, \qquad \nu = 1, 2, \dots \tag{18}$$

and $k_0, k_1, \dots$ is Markov chain on elements of the matrix L constructed by using an initial probability p_0 and a transition probability $p_{k_{\nu-1},k_\nu}$ for choosing the element $l_{k_{\nu-1},k_\nu}$ of the matrix L.

Now consider the following system of linear equations

$$Au = b \ , \tag{19}$$

where $A = [a_{ij}] \in I\!R^{m\times m}$ is a matrix, $b = (b_1, \dots, b_m)^T \in I\!R^{m\times 1}$ is an m-dimensional vector.

It is possible to choose a non singular matrix $M \in I\!R^{m\times m}$ such that

$$MA = I - L$$

($I \in I\!R^{m\times m}$ is the identity matrix), and

$$Mb = f, \quad f \in I\!R^{m\times 1}$$

and then (13) becomes

$$MAu = Mb,$$

i.e, (19). The last equation is equivalent to (14). If matrices M and A are both non-singular and L has its eigenvalues all inside the unit circle, then (17) becomes a *stationary linear iterative Monte Carlo algorithm.*

As a result the convergence of the Monte Carlo method depends on truncation error of (12).

2.1 Solving Linear Systems and Matrix Inversion. New Fast Algorithms

Consider a matrix L:

$$L = [l_{ij}], \quad L \in I\!R^{m\times m}$$

and a vector

$$f = (f_1, \dots, f_m)^T \in I\!R^{m\times 1}$$

The matrix L can be consider as a linear operator $L[I\!R^m \to I\!R^m]$, so that the linear transformation

$$Lf \in I\!R^{m\times 1}$$

defines a new vector in $I\!R^{m\times 1}$.

Since iterative Monte Carlo algorithms using the transformation (2.1) will be considered, the linear transformation (2.1) will be called *iteration*. The algebraic transformation (2.1) plays a fundamental role in iterative Monte Carlo algorithms.

Now consider the following two problems **Pi (i=1,2)** for the matrix L:

Problem P1. Evaluating the inner product

$$J(u) = (h, u) = \sum_{i=1}^{m} h_i u_i$$

of the solution $u \in I\!R^m$ of the linear algebraic system

$$Au = b,$$

where $A \in I\!R^{m\times m}$ is a given matrix; $b \in I\!R^{m\times 1}$ and $h \in I\!R^{m\times 1}$ are given vectors.

Clearly, as it was shown in Section 2 it is possible to choose a matrix $M \in I\!R^{m\times m}$ such that $MA = I - L$, where $I \in I\!R^{m\times m}$ is the identity matrix and $Mb = f$, $f \in I\!R^{m\times 1}$.

Then (19) becomes $u = Lu + f$.

It will be assumed that

$$(i) \qquad \begin{cases} 1. & \text{The matrices } M \text{ and } L \text{ are both non-singular;} \\ 2. & |\lambda(L)| < 1 \ \text{ for all eigenvalues } \ \lambda(L) \ \text{ of } \ L, \end{cases}$$

that is, all values $\lambda(L)$ for which

$$Lu = \lambda(L)u$$

is satisfied.

Problem P2. Inverting of matrices, i.e. evaluating of matrix

$$C = A^{-1},$$

where $A \in I\!R^{m\times m}$ is a given real matrix.

Assumed that the following conditions are fulfilled:

$$(ii) \qquad \begin{cases} 1. & \text{The matrix } A \text{ is non-singular;} \\ 2. & ||\lambda(A)| - 1| < 1 \text{ for all eigenvalues } \ \lambda(A) \ \text{ of } \ A. \end{cases}$$

Obviously, if the condition (i) is fulfilled, the solution of **the problem P1** can be obtained using the iterations (12).

For **the problem P2** the following *iterative* matrix:

$$L = I - A$$

can be constructed.

Since it is assumed that the condition (ii) is fulfilled, the inverse matrix $C = A^{-1}$ can be presented as $C = \sum_{i=0}^{\infty} L^i$.

Let $L = \{l_{ij}\}_{i,j=1}^m$ be a given matrix and $f = \{f_i\}_{i=1}^m$ and $h = \{h_i\}_{i=1}^m$ are vectors.

For the problems $\mathbf{P_i (i = 1, 2)}$ one can create a stochastic process using the matrix L and vectors f and h.

Consider an initial density vector $p = \{p_i\}_{i=1}^m \in I\!R^m$, such that $p_i \geq 0, i = 1, \ldots, m$ and $\sum_{i=1}^m p_i = 1$.

Consider also a transition density matrix $P = \{p_{ij}\}_{i,j=1}^m \in I\!R^{m \times m}$, such that $p_{ij} \geq 0,\ i, j = 1, \ldots, m$ and $\sum_{j=1}^m p_{ij} = 1$, for any $i = 1, \ldots, m$.

Consider the following Markov chain:

$$T_i = k_0 \to k_1 \to \ldots \to k_i. \tag{20}$$

where $k_j = 1, 2, \ldots, i$ for $j = 1, \ldots, i$ are natural random numbers.

The rules for constructing the chain (20) are:

$$Pr(k_0 = \alpha) = p_\alpha, \quad Pr(k_j = \beta | k_{j-1} = \alpha) = p_{\alpha\beta}. \tag{21}$$

Assume that

$$p = \{p_\alpha\}_{\alpha=1}^m \in \mathcal{P}_h, \quad P = \{p_{\alpha\beta}\}_{\alpha,\beta=1}^m \in \mathcal{P}_L,$$

where $\mathcal{P}_h$ and $\mathcal{P}_L$ are sets of *permissible* densities defined in [So73].

Now define the random variables Q_ν using the formula (18). One can see, that the random variables $Q_\nu, \nu = 1, \ldots, i$ can also be considered as weights on the Markov chain (21).

From all possible permissible densities we choose the following

$$p = \{p_\alpha\}_{\alpha=1}^m \in \mathcal{P}_h, \quad p_\alpha = \frac{|h_\alpha|}{\sum_{\alpha=1}^m |h_\alpha|};$$

$$P = \{p_{\alpha\beta}\}_{\alpha,\beta=1}^m \in \mathcal{P}_L, \quad p_{\alpha\beta} = \frac{|l_{\alpha\beta}|}{\sum_{\beta=1}^m |l_{\alpha\beta}|}, \alpha = 1, \ldots, m.$$

Such a choice of the initial density vector and the transition density matrix leads to an *Almost Optimal Monte Carlo* (MAO) algorithm. The initial density vector $p = \{p_\alpha\}_{\alpha=1}^m$ is called *almost optimal initial density vector* and the transition density matrix $P = \{p_{\alpha\beta}\}_{\alpha,\beta=1}^m$ is called *almost optimal density matrix* [Di91], [DT93a].

Let us consider Monte Carlo methods *with absorbing states*: instead of the finite random trajectory T_i in our algorithms we consider an infinite trajectory with a state coordinate $\delta_q (q = 1, 2, \ldots)$. Assume $\delta_q = 0$ if the trajectory is broken (absorbed) and $\delta_q = 1$ in other cases. Let $\Delta_q = \delta_0 \times \delta_1 \times \ldots \times \delta_q$.

So, $\Delta_q = 1$ up to the first break of the trajectory and $\Delta_q = 0$ after that.

It is easy to show, that under the conditions (i) and (ii), the following equalities are fulfilled:

$$E\{Q_i f_{k_i}\} = (h, L^i f), \quad i = 1, 2, \ldots;$$

$$E\{\sum_{i=0}^n Q_i f_{k_i}\} = (h, u), \quad (P1),$$

$$E\{\sum_{i|k_i=r'} Q_i\} = c_{rr'}, \quad (P2),$$

where $(i|k_i = r')$ means a summation only for weights Q_i for which $k_i = r'$ and $C = \{c_{rr'}\}_{r,r'=1}^n$.

To analyse the convergence of Monte Carlo algorithms consider the following functional equation

$$u - \lambda L u = f, \tag{22}$$

where λ is some parameter. Note that the matrices can be considered as linear operators. Define resolvent operator R_λ by the equation

$$I + \lambda R_\lambda = (I - \lambda L)^{-1},$$

where I is the *identity operator*.

Let $\lambda_1, \lambda_2, \ldots$ be the eigenvalues of the equation (22), where it is supposed that

$$|\lambda_1| \geq |\lambda_2| \geq \ldots$$

Monte Carlo algorithms are based on the representation

$$u = (I - \lambda L)^{-1} f = f + \lambda R_\lambda f,$$

where

$$R_\lambda = L + \lambda L^2 + \ldots, \tag{23}$$

The systematic error of (23) when μ terms are used is

$$r_s = O[(|\lambda|/|\lambda_1|)^{\mu+1} \mu^{\rho-1}], \tag{24}$$

where ρ is the multiplicity of the roots of λ_1.

From (24) is follows that when λ is approximately equal to λ_1 the sequence (23) and the corresponding Monte Carlo algorithm converges slowly. When $\lambda \geq \lambda_1$ the algorithm does not converge.

Obviously, the representation (23) can be used for $\lambda : |\lambda| < |\lambda_1|$ to achieve convergence.

Thus, there are two problems to consider.

Problem 1. How can the convergence of the Monte Carlo algorithm be accelerated when the corresponding Neumann series converges slowly,

and

Problem 2. How can a Monte Carlo algorithm be constructed when the sequence (23)

$$R_\lambda = L + \lambda L^2 + \ldots,$$

does not converge.

To answer these questions we apply a *mapping* of the spectral parameter λ in (22).

The method under consideration follows an approach which is similar to the method used by L. Kantorovich & G. Akilov [KA77] and K. Sabelfeld [Sa89] for integral equations. In [KA77] the mapping approach is used for solving some problems of numerical analysis.

To extend these results it is necessary to show that the mapping approach can be extended for any linear operators (including matrices).

Consider the problem of constructing the solution of (22) for $\lambda \in G$ and $\lambda \neq \lambda_k, k = 1, 2, \ldots$, where the domain G is a domain lying inside the definition domain of the $R_\lambda f$, such that all eigenvalues are outside of the domain G. In the neighbourhood of the point $\lambda = 0$ ($\lambda = 0 \in G$) the resolvent can be expressed by the series

$$R_\lambda f = \sum_{k=0}^{\infty} c_k \lambda^k,$$

where $c_k = L^{k+1} f$.

Consider the variable α in the unit circle on the complex plane $\Delta(|\alpha| < 1)$.

The function

$$\lambda = \psi(\alpha) = a_1\alpha + a_2\alpha^2 + \ldots,$$

maps the domain Δ into G. Now it is possible to use the following resolvent

$$R_{\psi(\alpha)} f = \sum_{j=0}^{\infty} b_n \alpha^j \ , \tag{25}$$

where $b_j = \sum_{k=1}^{j} d_k^{(j)} c_k$ and $d_k^{(j)} = \frac{1}{j!}\left[\frac{\partial^j}{\partial\alpha^j}[\psi(\alpha)]^k\right]_{\alpha=0}$.

It is clear, that the domain G can be chosen so that it will be possible to map the value $\lambda = \lambda_*$ into point $\alpha = \alpha_* = \psi^{-1}(\lambda_*)$ for which the sequence (25) converges; hence the solution of the functional equation (22) can be presented in the following form:

$$u = f + \lambda_* R_{\psi(\alpha_*)} f,$$

where the corresponding sequence for $R_{\psi(\alpha)} f$ converges absolutely and uniformly in the domain Δ .

This approach is also helpful when the sequence (23) converges slowly.

To apply this approach one needs some information about the spectrum of the linear operator (respectively, the matrix). Lat us assume, for example, that all eigenvalues λ_k are real and $\lambda_k \in (-\infty, -a]$, where $a > 0$. Consider a mapping for the case of interest ($\lambda = \lambda_* = 1$):

$$\lambda = \psi(\alpha) = \frac{4a\alpha}{(1-\alpha)^2}. \tag{26}$$

The sequence $R_{\psi(\alpha)} f$ for the mapping (26) converges absolutely and uniformly [KA77].

In Monte Carlo calculations we cut the sequence in (25) after μ terms

$$R_{\lambda_*} f \approx \sum_{k=1}^{\mu} b_k \alpha_k^k = \sum_{k=1}^{\mu} \alpha_*^k \sum_{i=1}^{k} d_i^{(k)} c_i = \sum_{k=1}^{\mu} g_k^{(\mu)} c_k, \tag{27}$$

where

$$g_k^{(\mu)} = (4a)^k \sum_{j=k}^{\mu} q_{k,j} \alpha_*^j. \tag{28}$$

It is easy to see that the coefficients $q_{k,j}$ are the following binomial coefficients

$$q_{k,j} = C_{k+j-1}^{2k-1}.$$

In order to calculate the iterations $c_k = L^{k+1} f$ a Monte Carlo method has to be used.

The mapping (26) creates the following Monte Carlo iteration process

$$\begin{aligned} u_0 &= f \\ u_1 &= 4aLu_0 \\ u_2 &= 4aLu_1 + 2u_1 \\ u_3 &= 4aLu_2 + 2u_2 - u_1 \\ u_j &= 4aLu_{j-1} + 2u_{j-1} - u_{j-2}, \quad j > 2. \end{aligned} \tag{29}$$

and from (29) we have

$$u^{(k)} = 4a\alpha L u^{(k-1)} + 2\alpha u^{(k-1)} - \alpha^2 u^{(k-2)} + f(1-\alpha^2), \quad k > 2.$$

Define the random variable $\theta^*[h]$

$$\theta^*[h] = \frac{h_{k_0}}{p_0} \sum_{\nu=0}^{\mu} g_\nu^{(\mu)} Q_\nu f_{k_\nu},$$

where $Q_0 = 1$, $g_0^{(\mu)} = 1$ and

$$Q_\nu = Q_{\nu-1} \frac{l_{k_{\nu-1},k_\nu}}{p_{k_{\nu-1},k_\nu}}, \qquad \nu = 1, 2, \dots,$$

($k_0, k_1, k_2, \dots$ is a Markov chain with initial density function p_{k_0} and transition density function $p_{k_{\nu-1},k_\nu}$) and coefficients $g_j^{(\mu)}$ are defined by (28) for $j \geq 1$.

The following theorem is proved:

Theorem 2.1 *Consider matrix L, whose Neumann series (23) does not converge. Let (26) be the required mapping, so that the presentation (27) exists. Then*

$$E\left\{ \lim_{m\to\infty} \frac{h_{k_0}}{p_0} \sum_{\nu=0}^{m} g_\nu^{(m)} Q_\nu f_{k_\nu} \right\} = (h, u).$$

Sketch of proof:

First consider the density of the Markov chain $k_0 \to k_1 \to \ldots \to k_i$ as a point in $m(i+1)$-dimensional Eucledian space $\mathbf{T_{i+1}} = \underbrace{I\!R^m \times \ldots \times I\!R^m}_{\mathbf{i+1}}$:

$$P\{k_0 = t_0, k_1 = t_1, \ldots, k_i = t_i\} = p_0 p_{t_0 t_1} p_{t_1 t_2} \ldots p_{t_{i-1} t_i}.$$

Now calculate the mathematical expectation of the random variable

$$\frac{h_{k_0}}{p_0} g_\nu^{(m)} Q_\nu f_{k_\nu}.$$

From the definition of the mathematical expectation it follows:

$$E\left\{\frac{h_{k_0}}{p_0} g_\nu^{(m)} Q_\nu f_{k\nu}\right\} = \sum_{t_0,\ldots,t_\nu=1}^{n} \frac{h_{t_0}}{p_0} g_\nu^{(m)} Q_\nu f_{t_\nu} p_0 p_{t_0 t_1} \ldots p_{t_{\nu-1} t_\nu} =$$

$$\sum_{t_0,\ldots,t_\nu=1}^{n} h_{t_0} l_{t_0 t_1} l_{t_1 t_2} \ldots l_{t_\nu - 1} t_\nu f_{t\nu} = (h, L^\nu f).$$

The existence and convergence of the sequence (28) ensures the following presentations:

$$\sum_{\nu=0}^{n} E\left|\frac{h_{k_0}}{p_0} g_\nu^{(m)} Q_\nu f_{k\nu}\right| = \sum_{\nu=0}^{n} (|h|, |L|^\nu |f|) = \left(|h|, \sum_{\nu=0}^{n} |L|^\nu |f|\right),$$

$$E\left\{\lim_{m\to\infty} \frac{h_{k_0}}{p_0} \sum_{\nu=0}^{m} g_\nu^{(m)} Q_\nu f_{k_\nu}\right\} =$$

$$\sum_{\nu=0}^{\infty} E\left\{\frac{h_{k_0}}{p_0} g_\nu^{(m)} Q_\nu f_{k_\nu}\right\} = \sum_{\nu=0}^{\infty} (h, L^\nu f) = (h, u). \quad \diamondsuit$$

This theorem permits the use of the random variable $\theta_\mu^*[h]$ for calculating the inner product (14).

For calculating one component of the solution , for example the "r"th component of u, we must to choose

$$h = e(r) = (0, ..., 0, 1, 0, ..., 0)^T,$$

where the one is in the "r"th position. It follows that $(h, u) = \sum_\alpha^m e_\alpha(r) u_\alpha = u_r$ and the corresponding Monte Carlo method is given by

$$u_r \approx \frac{1}{n} \sum_{s=1}^{n} \theta_\mu^*[e(r)]_s,$$

where n is the number of chains and

$$\theta_m^*[e(r)]_s = \sum_{\nu=0}^{m} g_\nu^{(m)} Q_\nu f_{k_\nu}; \quad Q_\nu = \frac{l_{r k_1} l_{k_1 k_2} \ldots l_{k_{\nu-1} k_\nu}}{p_{r k_1} p_{k_1 k_2} \ldots p_{k_{\nu-1} p_\nu}}.$$

To find the inverse $C = \{c_{rr'}\}_{r,r'=1}^{m}$ of some matrix A we must first compute the elements of the matrix

$$L = I - A, \tag{30}$$

where I is the identity matrix. Clearly the inverse matrix is given by $C = \sum_{i=0}^{\infty} L^i$, which converges if $\|L\| < 1$.

If the last condition is not fulfilled or if the corresponding Neumann series converges slowly we can use the same technique for accelerating the convergence of the algorithm.

Estimate the element $c_{rr'}$ of the inverse matrix C. Let the vector f given by (22) be the following unit vector $f_{r'} = e(r')$.

Theorem 2.2 *Consider matrix L, whose Neumann series (23) does not converge. Let (26) be the required mapping, so that presentation (27) exists. Then*

$$E\left\{\lim_{m\to\infty}\sum_{\nu=0}^{m} g_\nu^{(m)} \frac{l_{rk_1}l_{k_1k_2}\ldots l_{k_{\nu-1}k_\nu}}{p_{rk_1}p_{k_1k_2}\ldots p_{k_{\nu-1}p_\nu}} f_{r'}\right\} = c_{rr'}.$$

Sketch of proof: The proof is similar to the proof of Theorem 2.1, but in this case we need to consider an unit vector $e(r)$ instead of vector h and vector $e(r')$ instead of $f_{k\nu}$:

$$E\left\{\frac{e(r)}{1} g_\nu^{(m)} Q_\nu f_{k_\nu}\right\} = (e(r), L^\nu f) = (L^\nu f)_r.$$

So, in this case the "r"-th component of the solution is estimated:

$$u_r = \sum_{\nu=1}^{n} c_{ri} f_i$$

When $f_{r'} = e(r')$, one can get:

$$u_r = c_{rr'},$$

that is:

$$E\left\{\lim_{m\to\infty}\sum_{\nu=0}^{m} g_\nu^{(m)} \frac{l_{rk_1}l_{k_1k_2}\ldots l_{k_{\nu-1}k_\nu}}{p_{rk_1}p_{k_1k_2}\ldots p_{k_{\nu-1}k_\nu}} e(r')\right\} =$$

$$= \sum_{\nu=0}^{\infty} E\left\{g_\nu^{(m)} \frac{l_{rk_1}l_{k_1k_2}\ldots l_{k_{\nu-1}k_\nu}}{p_{rk_1}p_{k_1k_2}\ldots p_{k_{\nu-1}k_\nu}} e(r')\right\} = \sum_{\nu=0}^{\infty} (e(r), L^\nu e(r')) =$$

$$= \left(e(r), \sum_{\nu=0}^{\infty} L^\nu e(r')\right) = \sum_{i=1}^{n} c_{ri} e(r') = c_{rr'}. \quad \diamondsuit$$

Theorem 2.2 permits the use of the following Monte Carlo algorithm for calculating elements of the inverse matrix C:

$$c_{rr'} \approx \frac{1}{n}\sum_{s=1}^{m}\left[\sum_{(\nu|k_\nu=r')}^{\mu} g_\nu^{(\mu)}\frac{l_{rk_1}l_{k_1k_2}\ldots l_{k_{\nu-1}k_\nu}}{p_{rk_1}p_{k_1k_2}\ldots p_{k_{\nu-1}p_\nu}}\right]_s,$$

where $(\nu|k_\nu = r')$ means that only the variables

$$Q_\nu^{(\mu)} = g_\nu^{(\mu)}\frac{l_{rk_1}l_{k_1k_2}\ldots l_{k_{\nu-1}k_\nu}}{p_{rk_1}p_{k_1k_2}\ldots p_{k_{\nu-1}p_\nu}}$$

for which $k_\nu = r'$ are included in the sum (30).

Observe that since $Q_\nu^{(\mu)}$ is only contained in the corresponding sum for $r' = 1, 2, \ldots, m$ then the same set of n chains can be used to compute a single row of the inverse matrix , an important saving in computation which we make use of later.

2.2 Balancing of the errors

There are two errors in Monte Carlo algorithms: systematic and stochastic. It is clear that in order to obtain good results the stochastic error (the probable error) must be approximately equal to the systematic one, that is

$$r_n = O(r_s).$$

The problem of balancing the error is closely connected with the problem of obtaining an optimal ratio between the number of realizations n of the random variable and the mean value T of the number of steps in each random trajectory μ.

Let us consider the case when the algorithm is applied to **Problem 1**. Using the mapping procedure and a random variable, defined by (52) we accelerate the convergence of the algorithm proposed by Curtiss [Cu54],[Cu56]. This means that for a fixed number of steps μ

$$r_s(\mu) < r_s^{(C)}(\mu), \tag{31}$$

where $r_s^{(C)}(\mu)$ is the systematic error of the Curtiss algorithm and $r_s(\mu)$ is the systematic error of the algorithm under consideration. A similar inequality holds for the probable errors. Since $g_k^{(\mu)}$ from (52) it follows that

$$\sigma(\theta^*) < \sigma(\theta) \tag{32}$$

and thus

$$r_n(\sigma(\theta^*)) < r_n^{(C)}(\sigma\theta) \tag{33}$$

where $r_n^{(C)}$ is the probable error for the Curtiss algorithm.

Next consider the general error

$$R = r_n(\sigma) + r_s(\mu)$$

for matrix inversion by our Monte Carlo approach. Let R be fixed. Obviously from (31) and (32) it follows that there exist constants $c_s > 1$ and $c_n > 1$, such that

$$r_s^{(C)}(\mu) = c_s r_s, \;\; r_n^{(C)}(\sigma) = c_n r_n.$$

Since we are considering the problem of matrix inversion for a fixed general error R, we have

$$R = R^{(C)} = r_n^{(C)}(\sigma) + r_s^{(C)}(\mu) = c_n r_n(\sigma) + c_s r_s(\mu).$$

This expression shows that both parameters n and $T = E(\mu)$ or one of them, say n can be reduced. In fact

$$c\sigma(\theta)/n_c^{1/2} + r_s^{(C)}(\mu) = cc_n\sigma(\theta^*)/n_c^{1/2} + c_s r_s(\mu) = c\sigma(\theta^*)/n^{1/2} + r_s(\mu),$$

or

$$c\sigma(\theta^*)/n^{1/2} = cc_n\sigma(\theta^*)/n_c^{1/2} + (c_s - 1)r_s(\mu)$$

and

$$\frac{1}{n^{1/2}} = \frac{c_n}{n_c^{1/2}} + \frac{(c_s - 1)r_s(\mu)}{c\sigma(\theta^*)}, \tag{34}$$

where n_c is the number of realizations of the random variable for Curtiss' algorithm.

Denote by b the following strongly positive variable

$$b = \frac{(c_s - 1)r_s(\mu)}{c\sigma(\theta^*)} > 0. \tag{35}$$

From (34) and (35) we obtain:

$$n = \frac{n_c}{\left(c_n + b n_c^{1/2}\right)^2}. \tag{36}$$

The result (36) is an exact result, but from practical point of view it may be difficult to estimate $r_s(m)$ exactly. However, it is possible using (35) to obtain the following estimate for n

$$n < \frac{n_c}{c_n^2}.$$

This last result shows that for the algorithm under consideration the number of 0 n can be at least c_n^2 times less than the number of realizations n_c of the existing algorithm. Thus it is seen that there are a number of ways in which we can control the parameters of the fixed size array in [MAD94] Below we explore some of these possibilities and develop some comparisons.

3 Monte Carlo Algorithms for BVP

Let $G \subset \mathbb{R}^d$ be a bounded domain with a boundary ∂G. The following notations are used:

$x = (x_1, x_2, \ldots, x_d)$ is a point in $\mathbb{R}^d$;

$D^\alpha = D_1^{\alpha_1} D_2^{\alpha_2} \ldots D_d^{\alpha_d}$ is an $|\alpha| = \alpha_1 + \alpha_2 + \ldots + \alpha_d$ derivative, where $D_i = \partial/\partial x_i,\ i = 1, \ldots, d$ and $C^k(\bar{G})$ is a space of functions $u(x)$ continuous on $\bar{G}$ such that $D^\alpha u$ exists in G and admits a continuous extension on $\bar{G}$ for every $\alpha : |\alpha| \leq k$.

We consider the linear boundary value problem

$$Lu \equiv \sum_{|\alpha| \leq 2m} a_\alpha(x) D^\alpha u(x) = -f(x), \qquad x \in G \tag{37}$$

$$u(x) = \varphi(x), \qquad x \in \partial G\,, \tag{38}$$

where L is an arbitrary linear elliptic operator in $I\!R^d$ of order $2m$, $a_\alpha(x) \in C^\infty(I\!R^n)$ and the function $f(x)$ belongs to the Banach space $\mathbf{X}(G)$.

Assume that $f(x)$, $\varphi(x)$, and the boundary ∂G satisfy conditions ensuring that the solution of the problem (37), (38) exists and is unique [Mi83], [Mi55].

We shall study Monte Carlo algorithms for calculating linear functionals of the solution of the problem (37), (38)

$$J(u) = (h, u) = \int_G u(x)h(x)dx, \tag{39}$$

where $h \in \mathbf{X}^*(G)$ ($\mathbf{X}^*(G)$ is the conjugate functional space to $\mathbf{X}(G)$).

For many applications $\mathbf{X} = L_1$ and thus $\mathbf{X}^* = L^\infty$, or $\mathbf{X} = L_2$, $\mathbf{X}^* = L_2$.

There are two approaches for calculating (39). The first approach uses a discretisation of the problem (37, 38) on a mesh and solves the resulting linear algebraic system, which approximates the original problem (37, 38). This is the so-called *grid Monte Carlo algorithm*, or *grid walk algorithm*. The second approach (grid-free approach) uses an integral representation for the problem (37, 38).

Let $f = f(x)$ and $u_0 = u(x_0)$ be real-valued functions, defined in $G \subset I\!R^d$ and $L = L(u)$ be a linear operator defined on the space of all real-valued functions u whose arguments belong to G.

3.1 Grid Monte Carlo Algorithm

Consider a regular mesh (lattice) with step-size h in $I\!R^d$. Let G_h be the set of all inner mesh points ($\gamma \in G_h$ if and only if $\gamma \in G$); ∂G_h be the set of all "boundary" mesh points ($\gamma \in \partial G_h$ if there exists a neighbouring mesh point γ^* which does not belong to $I\!R^d \setminus \bar{G}$) and u_h be a function defined on a set of mesh points (a mesh function).

The differential operator L at the mesh point $x_i \in G_h$ is approximated by a difference operator L_h as follows:

$$(L_h u_h)_i = \sum_{x_j \in P_k(x_i)} a_h(x_i, x_j) u_h(x_j)\ , \tag{40}$$

where $a_h(x_i, x_j)$ are coefficients; and $P_k(x_i)$ is a set of mesh points with center in $x_i \in G_h$ called a scheme.

Since L is a linear differential operator, after the discretisation of (40), the following system of linear equation arises:

$$Au = b\ ,$$

where $b = (b_1, \ldots, b_m) \in I\!R^{m\times 1}$ is an m-dimensional vector and $A \in I\!R^{m\times m}$ is an $m \times m$-dimensional matrix.

3.2 Grid-Free Monte Carlo Algorithms

The grid-free Monte Carlo approach under consideration is based on use of local integral representation of the solution. In this case the Green's function for standard domains, lying inside the domain G (for example - ball, sphere, ellipsoid) is used.

Consider the elliptic boundary value problem:

$$Mu = -\phi(x), \ x \in G, \ G \subset I\!R^3 \tag{41}$$

$$u = \psi(x), \ x \in \partial G, \tag{42}$$

where

$$M = \sum_{i=1}^{3} \left(\frac{\partial^2}{\partial x_i^2} + b_i(x) \frac{\partial}{\partial x_i} \right) + c(x).$$

Define the class of domains $\mathbf{A}^{(k,\lambda)}$.

Definition 3.1 *The domain G belongs to the class $\mathbf{A}^{(k,\lambda)}$ if for any point $x \in \partial G$ (from the boundary ∂G) the boundary ∂G can be presented as a function $z_3 = \sigma(z_1, z_2)$ in the neighbourhood of x for which $\sigma^{(k)}(z_1, z_2) \in \mathbf{C}^{(0,\lambda)}$, i.e.*

$$|\sigma^{(k)}(y) - \sigma^{(k)}(y')| \leq N|y - y'|^{\lambda}$$

where the vectors $y \equiv (z_1, z_2)$ and $y' \equiv (z_1', z_2')$ are 2-dimensional vectors, N is constant and $\lambda \in (0, 1]$.

If in the closed domain $\bar{G} \in \mathbf{A}^{(1,\lambda)}$ the coefficients of the operator M satisfy the conditions $b_j, \ c(x) \in \mathbf{C}^{(0,\lambda)}(\bar{G}), \ c(x) \leq 0$ and $\phi \in \mathbf{C}^{(0,\lambda)}(G) \cap \mathbf{C}(\bar{G}), \ \psi \in \mathbf{C}(\partial G)$, the problem (41), (42) has an unique solution $u(x)$ in $\mathbf{C}^2(G) \cap \mathbf{C}(\bar{G})$.

The conditions for uniqueness of a solution can be found in ([Mi83], p. 179, [Bi82], p. 79).

We obtain an integral representation of the solution $u(x)$. This representation allows to use the random variable for calculation the functional (39).

3.3 Local Integral Representation

We have to estimate the functional (39) by means of grid-free Monte Carlo approach. This approach is based on the use of a local integral representation of the solution $u(x)$ in the problem (41), (42). The representation uses the Green's function approach for standard domains, lying inside the domain G.

The initial step in studying the grid-free Monte Carlo approach is obtaining of an integral representation of the solution in the form:

$$u(x) = \int_{B(x)} k(x, y) u(y) dy + f(x) \tag{43}$$

assuming that a representation exists.

The iterative Monte Carlo process converges when the condition

$$\| K(u) \|_{L_1} = \max_{x\in G} \int_G | k(x,y) | \, dy \le q < 1 \tag{44}$$

holds.

For the existence of the integral representation, (43) might be obtained using the result of C. Miranda [Mi55] taking into consideration that the domain $B(x)$ belongs to the space $\mathbf{A}^{(1,\lambda)}$ and that the operator M is of elliptic type.

We seek a representation of the integral kernel $k(x,y)$ using Levy's function and the adjoint operator M^* for the initial differential operator M.

The Levy's function for the problem (41), (42) is

$$L_p(y,x) = \mu_p(R) \int_r^R (1/r - 1/\rho) p(\rho) d\rho, \quad r \le R \tag{45}$$

where the following notations are used:

$p(\rho)$ is a density function;

$$r = | x - y | = \left(\sum_{i=1}^{3} (x_i - y_i)^2 \right)^{1/2};$$

$$\mu_p(R) = [4\pi q_p(R)]^{-1}; \;\; q_p(R) = \int_0^R p(\rho) d\rho.$$

It is clear that Levy's function $L_p(y,x)$, and the parameters $q_p(R)$ and $\mu_p(R)$ depend on the choice of the density function $p(\rho)$. In fact, the equality (45) defines a family of functions.

We seek a choice of $p(\rho)$ which leads to a representation of type (43). Moreover, the kernel of the integral transform should be a transition density function, i.e. $k(x,y) \ge 0$.

From an algorithmic point of view the domain $B(x)$ must be chosen in such a way that the coordinates of the boundary points $y \in \partial B(x)$ could be easily calculated.

Denote by $B(x)$ the ball:

$$B(x) = B_R(x) = \{ y : r = | y - x | \le R(x) \} \tag{46}$$

where $R(x)$ is the radius of the ball.

For Levy's function $L_p(y,x)$ the following representation holds (see, [Mi55]):

$$u(x) = \int_{T(x)} \left(u(y) M_y^* L_p(y,x) + L_p(y,x)\phi(y) \right) dy +$$

$$+ \int_{\partial T(x)} \sum_{i=1}^{3} n_i \left[\left(\frac{L_p(y,x)\partial u(y)}{\partial y_i} - \frac{u(y)\partial L_p(y,x)}{\partial y_i} \right) - b_i(y) u(y) L_p(y,x) \right] d_y S \,, \tag{47}$$

where $\mathbf{n} \equiv (n_1, n_2, n_3)$ is the exterior normal to the boundary $\partial T(x)$.

Formula (47) holds for any domain $T(x) \in \mathbf{A}^{(1,\lambda)}$ contained in G.

Obviously, $B(x) \in \mathbf{A}^{(1,\lambda)}$ and therefore for every ball lying inside the domain G the representation (47) holds.

Now we express the solution $u(x)$ by Green's function $G(x, y)$. It is known, that Green's function is a solution of the problem:

$$M_y^* G(x, y) = -\delta(x - y), \ y \in G \setminus \partial G \setminus \{x\},$$
$$G(x, y) = 0, \quad y \in \partial G, \ x \in G \setminus \partial G$$

Green's function is Levy's function, $L_p(y, x)$, for which (41), (42) hold.

Under the condition $L_p(y, x) = G(x, y)$ from (47) it is possible to get the integral representation:

$$u(x) = \int_{B(x)} G(x, y) f(y) dy - \int_{\partial B(x)} \sum_{i=1}^{3} n_i \frac{\partial G(x, y)}{\partial y_i} u(y) d_y S. \tag{48}$$

The representation (48) is the basis for the Monte Carlo algorithm.

For achieving this aim it is necessary to have a non-negative integral kernel. The next lemma shows that it is possible to construct the Levy's function choosing the density $p(\rho)$ such that $M_y^* L_p(y, x)$ is non-negative in $B(x)$ and such that $L_p(y, x)$ and its derivatives vanish on $\partial B(x)$, i.e.

$$L_p(y, x) = \partial L_p(y, x)/\partial y_i = 0 \ \text{ for } \ y \in \partial B(x), \ \ i = 1, 2, 3.$$

Lemma 3.1 *The conditions*

$$M_y^* L_p(y, x) \geq 0 \ \textit{ for any } \ y \in B(x)$$

and

$$L_p(y, x) = \partial L_p(y, x)/\partial y_i = 0, \ \textit{for any} \ \ y \in \partial B(x), \ i = 1, 2, 3$$

are satisfied for

$$p(r) = e^{-kr},$$

where

$$k \geq \max_{x \in G} \mid \mathbf{b}(x) \mid + R \max_{x \in G} \mid c(x) \mid \tag{49}$$

and R is the radius of the maximal ball $B(x) \subset \bar{G}$.

It follows that the representation (43) can be written in the form:

$$u(x) = \int_{B(x)} M_y^* L_p(y, x) u(y) dy + \int_{B(x)} L_p(y, x) \phi(y) dy. \tag{50}$$

The last representation allows to construct an unbiased estimate for the solution of our problem.

3.3.1 Monte Carlo Algorithms

Simple numerical examples for performing grid and grid-free Monte Carlo algorithms will now be considered.

Let the operator L in the equation (37) be the Laplacian :

$$L = \Delta.$$

Using a regular discretisation with a step–size h equation (37) is approximated by the following difference equation

$$\Delta_h^{(d)} u = -f_h \tag{51}$$

or solved for the i-th point $i = (i_1, \ldots, i_d)$

$$u_i = L_h u + \frac{h^2}{2d} f_i,$$

where $\Delta_h^{(d)}$ is the Laplace difference operator, and L_h is an averaging operator. For example, the operator L_h in $I\!R^2$ is

$$L_h u = \frac{1}{4}[u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}] = \frac{1}{4}\Lambda_1(i,j)$$

and then (51) becomes

$$u_{ij} = \frac{1}{4}\Lambda_1(i,j) + \frac{h^2}{4} f_{i,j}. \tag{52}$$

The matrix form of equations (52) has only $2d$ non-zero elements in every row and they all are equal to $1/2d$.

The grid Monte Carlo algorithm for solving (52) consists in simulating a Markov chain with initial density p_0 which is *permissible* to the vector h from the integral (39), and the probability $p_{\alpha\beta}$ for the transition from the point α to the next point β is equal to $\frac{1}{2d}$ if the point is inside of the domain, that is $((x_1)_\alpha, (x_2)_\beta) \in G_h$, and $p_{\alpha\beta} = 0$ for boundary points (the boundary is an absorbing barrier for the process). Then the random variable whose mathematical expectation coincides with the solution of the problem is:

$$\theta = \frac{h^2}{2d} \sum_{i=1}^{i^*-1} f_i + \varphi_{i^*},$$

where f_i are values of the function f in the points of the Markov chain, and i^* is the point where Markov chain reaches the boundary ∂G_h

The well known grid algorithm (see, for example, [Sh64] can be described in pseudo-code notation:

Start at the grid point $(x_{1,0}, x_{2,0})$, i.e. $(x_1, x_2) := (x_{1,0}, x_{2,0})$
While (x_1, x_2) is not at the boundary

Move to a neighboring point $(x_1', x_2') \in \{(x_1 - h, x_2),$
$(x_1 + h, x_2), (x_1, x_2 - h), (x_1, x_2 + h)\}$ (i.e.$(x_1, x_2) := (x_1', x_2')$)
such that each neighbouring is selected with
the same probability $p = 1/4$

Let (x_1^*, x_2^*) be the final point at the boundary. Then, the searched random variable is:

$$\theta := u(x_1^*, x_2^*).$$

From the theory, presented in Section 2 , it follows $u(x_{1,0}, x_{2,0}) = E\theta$, which is the desired property. In order to compute $E\theta$, we start n Markov processes of the above kind, delivering n realizations $\theta_1, \ldots, \theta_n$ of the random variable θ and approximate the solution by their mean as described above.

Now consider the *grid-free Monte Carlo algorithm* based on the local integral representation of the problem.

First, let us describe the selection algorithm in general case. Suppose that $v_1(x)$ and $v_2(x)$ are given functions, $0 \leq v_1(x) \leq v_2(x)$ and

$$\int_G v_1(x)dx = V_1 < \infty, \quad \int_G v_2(x)dx = V_2 < \infty,$$

where $G \subset I\!\!R^3$.

Consider an algorithm for simulation of the random variable with density function $v_2(x)/V_2$ and simulate other random variable with the density function $v_1(x)/V_1$. It is necessary to give a realization ξ of the random variable with density $v_2(x)/V_2$ and an independent realization γ of the random variable uniformly distributed in $(0, 1)$, as well as to check the inequality $\gamma v_2(x) \leq v_1(x)$. If the last inequality holds, ξ is the needed realization. Otherwise, the process have to be repeated. The efficiency of the selection algorithm is measured by $E = V_1/V_2$.

A local integral representation (50) for the boundary value problem (41), (42) is obtained. Compare (43) with (50) one can get

$$k(x,y) = \begin{cases} M_y^* L_p(y,x), & \text{when} \quad x \in G \setminus \partial G, \\ 0, & \text{when} \quad x \in \partial G, \end{cases}$$

and

$$f(x) = \begin{cases} \int_{B(x)} L_p(y,x)\phi(y)dy & \text{when} \quad x \in G \setminus \partial G, \\ \psi(x), & \text{when} \quad x \in \partial G. \end{cases}$$

The Monte Carlo procedure for solving this problem can be defined as a *ball process*. To ensure the convergence of the process, we introduce the ε-strip of the boundary, i.e.

$$\partial G_\varepsilon = \{x \in G : B(x) = B_\varepsilon(x)\}, \text{ where } B_\varepsilon(x) = \{y : r =\mid y - x \mid \leq \varepsilon\}.$$

Consider a transition density function

$$p(x, y) = k(x, y) = M_y^* L_p(y, x) \geq 0. \tag{53}$$

This transition density function defines a Markov chain $\xi_1, \xi_2, \ldots, \xi_i$ such that every point ξ_j, $j = 1, \ldots, i-1$ is chosen on the maximal ball $B(x_{j-1})$, lying in G in accordance with the density (53). The Markov chain stops when it reaches ∂G_ε. So, $\xi_i \in \partial G_\varepsilon$.

Let us consider the random variable

$$\theta[\xi_0] = \sum_{j=0}^{i} Q_j \int_{B(\xi_j)} L_p(y, \xi_j) f(y) dy + \varphi(\xi_i)$$

where

$$Q_0 = 1 \tag{54}$$

$$Q_j = Q_{j-1} M_y^* L_p(\xi_j, \xi_{j-1}) / p(\xi_{j-1}, \xi_j), j = 1, 2, \ldots, i, \tag{55}$$

$\varphi(\xi_i)$ is the value of the boundary function at the last point of the Markov chain ξ_i.

It is easy to see that the solution of the problem in the point ξ_0 can be presented as

$$u(\xi_0) = E\theta[\xi_0]. \tag{56}$$

To ensure the convergence of the process we consider the next estimate:

$$\int \int k(x, y) k(y, z) dy dz < \int \delta(y - z) \int \delta(z - y) dz dy = \int \delta(y - x) dy < 1 - \frac{\varepsilon^2}{4R_m^2},$$

where $k(x, y) \geq 0$ is defined by (53) and R_m is the supremum of all radii of the spheres lying in G.

The above estimate ensures the convergence of the Neumann series and, therefore of the process (55) as well.

Obviously, all non-zero values of Q_j are equal to 1 and the problem consists in simulating a Markov chain with a transition density function $p(x, y)$ in the form (53). Thus, the problem of calculating $u(\xi_0)$ is reduced to estimation the expectation (56). The computational problem then becomes one of calculating repeated realizations of $\theta[\xi_0]$ and combining them into an appropriate statistical estimator of $u(\xi_0)$.

As approximate value of $u(\xi_0)$ is set up

$$\theta_n = 1/n \sum_{s=1}^{n} \{\theta[\xi_0]\}_s,$$

where $\{\theta[\xi_0]\}_s$ is the s-th realization of the random variable $\theta[\xi_0]$ on a Markov chain with initial point ξ_0 and transition density function (53).

The direct simulation of a random variable with the stationary density function $p(x, y)$ is unsuitable since the complexity of the expression for $M_y^* L(y, x)$ would sharply increase the algorithm's computational complexity. In this case it is advisable to use the *selection algorithm.*

Denote by $p_0(x,y)$ the transition density function of the Markov chain $M_y^* L_p$ with $c(x) \equiv 0$.

It is easy to see, that $p(x,y) \leq p_0(x,y)$.
The function $p_0(x,y)$ satisfies the condition for a transition density function of the Markov chain:

$$\int_{B(x)} p_0(x,y)dy = 1. \tag{57}$$

The function $p_0(x,y)$ can be expressed as

$$p_0(r,\mathbf{w}) = \mu_p(R)\sin\theta \left[p(r) + \sum_{i=1}^{3} b_i(x+r\mathbf{w})w_i \right] \int_r^R p(\rho)d\rho,$$

or

$$p_0(r,\mathbf{w}) = \frac{\sin\theta}{4\pi q_p(R)} \left[p(r) + \sum_{i=1}^{3} b_i(x+r\mathbf{w})w_i \right] \int_r^R p(\rho)d\rho.$$

Here $\mathbf{w} \equiv (w_1, w_2, w_3)$ is an unique isotropic vector in $I\!R^3$, where $w_1 = \sin\theta\cos\varphi$, $w_2 = \sin\theta\sin\varphi$ and $w_3 = \cos\theta$.

Now write $p_0(r,\mathbf{w})$ in the following form:

$$p_0(r,\mathbf{w}) = p_0(r)p_0(\mathbf{w}/r),$$

where

$$p_0(r) = \frac{p(r)}{q_p(R)} = \frac{ke^{-kr}}{1-e^{-kR}}$$

is a density function and

$$p_0(\mathbf{w}/r) = \frac{\sin\theta}{4\pi} \left[1 + \frac{\mid \mathbf{b}(x+r\mathbf{w}) \mid \cos(\mathbf{b},\mathbf{w})}{p(r)} \int_r^R p(\rho)d\rho \right]$$

is a conditional density function.

In [ENS84] it is proved that $E \geq \frac{1}{2}$ for the same density function and for the boundary value problem in $I\!R^d (d \geq 2)$.

In [Di89] a majorant function $h_r(\mathbf{w})$ for $p_0(\mathbf{w}/r)$ was found and the following theoretical result for the algorithm efficiency of the selection grid-free Monte Carlo algorithm was proved:

$$E \geq \frac{1+\alpha}{2+\alpha}, \tag{58}$$

where

$$\alpha = \frac{\max_{x\in G} \mid c(x) \mid R}{\max_{x \in G} \mid \mathbf{b}(x) \mid},$$

and R is the radius of the maximal sphere lying inside G.

Now, we can prove the following result:

Theorem 3.1 *For the algorithm efficiency of the selection grid-free Monte Carlo algorithm the inequality:*

$$E \geq \frac{1+\alpha}{2+\alpha-\varepsilon_R}, \quad 0 < \varepsilon_R = \frac{1}{e^{kR}} < 1,$$

holds, when the majorant function

$$h_r(\mathbf{w}) = \frac{\sin\theta}{4\pi}\left[1 + \frac{\max_{x\in G} \mid \mathbf{b}(x) \mid}{p(r)} \int_r^R p(\rho)d\rho\right]$$

is used.

The proof of this result is given in [DG97].

Denote by $\bar{p}(x,y)$ the following function:

$$\bar{p}(x,y) = \frac{p(x,y)}{V}, \text{ where } \int_{B(x)} p(x,y)dy = V < 1,$$

This is a density function in the case when $c(y) \neq 0$.

The function $p(x,y)$ can be expressed in spherical coordinates as:

$$p(r,\mathbf{w}) = \frac{\sin\theta}{4\pi q_p(R)} \times$$

$$\times \left[p(r) + \left(\sum_{i=1}^{3} b_i(x+r\mathbf{w})w_i + c(x+r\mathbf{w})r\right)\int_r^R p(\rho)d\rho - c(x+r\mathbf{w})r^2 \int_r^R \frac{p(\rho)}{\rho}d\rho\right].$$

The following inequalities hold:

$$p(r,\mathbf{w}) \leq p_0(r,\mathbf{w}) \leq \frac{p(r)}{q_p(R)} h_r(\mathbf{w}) = \tag{59}$$

$$= \frac{\sin\theta\, p(r)}{4\pi q_p(R)}\left[1 + \frac{\max_{x\in G} \mid \mathbf{b}(x) \mid}{p(r)} \int_r^R p(\rho)d\rho\right] \equiv h(r,\mathbf{w}).$$

It is easy to prove that in the case when $h(r,\mathbf{w})$ is a majorant of the function $p(r,\mathbf{w})$ the efficiency of the selection method is

$$E \geq V \frac{1+\alpha}{2+\alpha-\varepsilon_R}.$$

This estimation follows from Theorem 3.1 and (59). Clearly, the efficiency E depends on the norm of the kernel $k(x,y)$, because $p(x,y) = k(x,y)$.

In the selection algorithm it is necessary to simulate a random variable η with a density

$$\bar{p}_r(\mathbf{w}) =$$

$$= 1 + \left[\frac{\mid \mathbf{b}(x + r\mathbf{w}) \mid \cos(\mathbf{b}, \mathbf{w}) + c(x + r\mathbf{w})r}{p(r)} \right] \times$$

$$\times \int_r^R p(\rho) d\rho - \frac{c(x + r\mathbf{w})r^2}{p(r)} \int_r^R \frac{p(\rho)}{\rho} d\rho.$$

Since

$$\bar{p}_r(\mathbf{w}) \leq 1 + \frac{B}{p(r)} \int_r^R p(\rho) d\rho = h(r)$$

the function $h(r)$ is a majorant for the selection method.

Here a Monte Carlo algorithm for the selection method is described:

Consider a point $x \in G$ with the initial density $p(x)$. Suppose that $p(x)$ is permissible to $g(x)$.

Algorithm 3.1 *:*

Grid-free Monte Carlo Algorithm

1. **Calculate** *the radius $R(x)$ of the maximal sphere lying inside G and having center x.*

2. **Calculate** *a realization r of the random variable τ with the density*

$$\frac{p(r)}{q_p(R)} = \frac{ke^{-kr}}{1 - e^{-kR}}. \tag{60}$$

3. **Calculate** *the function*

$$h(r) = 1 + \frac{B}{p(r)} \int_r^R p(\rho) d\rho = 1 + \frac{B}{k}(1 - e^{-k(R-r)}).$$

4. **Construct** *independent realizations $\mathbf{w}_j$ of an unique isotropic vector in $\mathbb{R}^3$.*

5. **Construct** *independent realizations γ_j of an uniformly distributed random variable in the interval $[0, 1]$.*

6. **Calculate** *the parameter j_0, given by*

$$j_0 = \min\{j : h(r)\gamma_j \leq \bar{p}_r(\mathbf{w}_j)\},$$

is calculated and **stop** *the execution of the steps 4 and 5. The random vector $\mathbf{w}_{j_0}$ has the density $\bar{p}_r(\mathbf{w})$.*

7. **Calculate** *the random point y, with a density $\bar{p}_r(\mathbf{w})$, using the following formula:*

$$y = x + r\mathbf{w}_{j_0}.$$

The value $r = \mid y - x \mid$ is the radius of the sphere lying inside G and having center at x.

8. **Stop** *the algorithm when the random process reaches the ε - strip ∂G_ε, i.e. $y \in \partial G_\varepsilon$. The random variable is calculated.* **If** *$y \bar{\in} \partial G_\varepsilon$* **then** *the algorithm has to be repeated for $x = y$.*

It is clear that the algorithmic efficiency depends on the expectation of the position of the point y. The location of y depends of the random variable τ with a density (60). When the random point is near to the boundary of the ball, the process goes to the boundary of the domain ∂G_ε quickly. So, it will be important to have an estimation of the mathematical expectation of τ.

One can calculate that

$$E\tau = \int_0^R r \frac{p(r)}{q_p(R)} dr = \int_0^R \frac{rke^{-kr}}{1 - e^{-kR}} dr = \frac{1}{k} + \frac{R}{1 - e^{kR}}.$$

Obviously, the sequentional algorithmic efficiency depends of the product kR (where R is the radius of the maximal ball, lying inside of the domain G for the starting point of the random process). Therefore, the computational results given in the next section are performed for different values of the product kR.

3.4 Parallel Implementation of the Grid-free Algorithm and numerical results

In the previous subsection a general description of the Monte Carlo algorithm for selection method has been provided. Note that, in the case of an implementation on a sequential computer, all the steps of the algorithm and all the trajectory are executed iteratively, whereas on a parallel computer each trajectory can be carried concurrently.

Example. A numerical examples are considered. The example deals with the following problem

$$\sum_{i=1}^{3} \left(\frac{\partial^2 u}{\partial x_i^2} + b_i(x) \frac{\partial u}{\partial x_i} \right) + c(x)u = 0, \ \text{ in } G = [0,1]^3.$$

Note that the cube $G = [0,1]^3$ does not belong to the $\mathbf{A}^{(1,\lambda)}$, but this restriction is not important for our algorithm since an ε-strip of the domain G is considered. In fact now we consider another domain G_ε which belongs to the class $\mathbf{A}^{(1,\lambda)}$.

The boundary conditions for the example are:

$$u(x_1, x_2, x_3) = e^{a_1 x_1 + a_2 x_2 + a_3 x_3}\ , \ (x_1, x_2, x_3) \in \partial G.$$

In our tests

$$b_1(x) = a_2 a_3 (x_2 - x_3),\ b_2(x) = a_3 a_1 (x_3 - x_1),\ b_3(x) = a_1 a_2 (x_1 - x_2)$$

(thus, the condition $div\, \mathbf{b}(x) = 0$ is valid) and

$$c(x) = -(a_1^2 + a_2^2 + a_3^2),$$

where a_1 , a_2 , a_3 are parameters.

The problem is solved using selection grid-free Monte Carlo algorithm.

We consider three cases for the coefficients:

- the first case when

$$a_1 = 0.25\,,\ a_2 = 0.25\,,\ a_3 = 0.25\ \text{ and }\ k * R = 0.101;$$

- the second case when

$$a_1 = 0.5\,,\ a_2 = 0.5\,,\ a_3 = 0.5\ \text{ and }\ k * R = 0.40401;$$

- the third case

$$a_1 = -1\,,\ a_2 = -1\,,\ a_3 = -1\ \text{ and }\ k * R = 1.61603.$$

Four different ε-strip are used:

$$\varepsilon = 0.01,\ 0.05,\ 0.1,\ 0.3.$$

$$h(x) = \delta[(x_1 - 1/2), (x_2 - 1/2), (x_1 - 1/2)].$$

The efficiency of the selection grid-free Monte Carlo does not depend on the number of trajectories (see, Table 3.4). The result of selection efficiency confirms our corresponding theoretical result.

$$h(x) = \delta[(x_1 - 1/4), (x_2 - 1/4), (x_1 - 1/4)],$$

$$h(x) = \delta[(x_1 - 1/2), (x_2 - 1/2), (x_1 - 1/2)],$$

respectively.
We investigate the parallel efficiency for the following values of the coefficients:

$$a_1 = 1\,,\ a_2 = -0.5\,,\ a_3 = -0.5\ \text{ and }\ \varepsilon - \text{strip} = 0.01\,,\ 0.05\,,\ 0.15.$$

3.4.1 Conclusion

- An iterative Monte Carlo algorithm used Green's function is presented and studied. It is proved that the integral transformation kernel in local integral presentation can be used as transition density function in the Markov chain. An algorithm "ball process" is presented. This algorithm is a grid-free Monte Carlo algorithm and uses the so-called selection .

- One of the advantages of the grid-free Monte Carlo algorithm is that it has the rate of convergence ($|log\ r_n|/r_n^2$) (where r_n is the statistical error) which is better than the rate r_n^{-3} of the grid algorithm. This means that the same error can be reached for a smaller number of trajectories.

- It is preferable to use the selection method when it is difficult to calculate the realizations of the random variable directly.

Table 1: **Selection efficiency and number of the steps to the boundary domain.**

$Epsilon\ strip$	$k * R$	$No\ of\ Steps$	$Selection\ efficiency$
0.01	0.101	$36-37$	0.99
0.01	0.40401	$35-36$	0.97123
0.01	1.61603	$43-44$	0.91071
0.05	0.101	$17-18$	0.99
0.05	0.40401	$17-18$	0.9596
0.05	1.61603	$20-21$	0.85829
0.10	0.101	$8-9$	0.9887
0.10	0.40401	$9-10$	0.95371
0.10	1.61603	$12-13$	0.83596
0.30	0.101	$1-2$	0.97
0.30	0.40401	$2-3$	0.92583
0.30	1.61603	$2-3$	0.75561

Remarks:

1. The number, which starts generator is 653;
2. The number of realizations of the random ball process is 600.

- The studied algorithm have high parallel efficiency. It is easily programmable and parallelizable.
- The tests performed show that Monte Carlo algorithms can be efficiency implemented on MIMD-machines.

4 Applications: Sensitivity Analysis in Environmental Mathematics

Here a special Monte Carlo sensitivity technique to study air-pollution transport over Europe is presented.

First, the developed technique is applied on the box-model in order to study the sensitivity of the concentrations of some important pollutants (like NO_2 and O_3). It is shown that the most important parameter (from the point of view of sensitivity simulations) is the rate constant of the reaction producing NO_2 from O_3 and NO. The results are sensitive to small variances of the rate constant.

Second, the developed Monte Carlo simulation technique is apply to the **Danish Eulerian Model**. For running the model to get realistic results for a real-live scenario of air-pollution transport the vector machine CRAY Y-MP C92A is used.

It is shown that the results of the real-live modeling of air-pollution transport are not equal sensitive to different parameters used in the model as input parameters. There are some parameters (like the rate constant of the reaction producing NO_2 from O_3 and NO), which are very important since the results are sensitive to the small changes of the values of these parameters. In this sense the Monte Carlo sensitivity simulation could by used as a special "advisor" to physicists, because this simulation permits to find how accurate have to be measured the input parameters, as well as how strong theory is needed to describe some of the processes of air-pollution transport.

High pollution levels may lead to the destruction of eco-systems and may cause damages on plants, animals and humans. Therefore, the pollution levels must be studied carefully. It is necessary to find out whether the pollution levels are under some critical levels and, if they are, to find reliable control strategies to keep them there. These tasks can successfully be solved by developing and using reliable mathematical models for studying different pollution phenomena. These models must satisfy several important requirements:

1. The mathematical models must be defined on large space domains, because the long range transport of air pollution is an important environmental phenomenon and high pollution levels are not limited to the areas where the high emission sources are located.

2. All relevant physical and chemical processes must be adequately described in the models used.

3. Enormous files of input data (both meteorological data and emission data) are needed.

4. Also the output files are normally very big, and fast visualization tools must be used in order to represent the trends and tendencies, hidden behind many megabytes (or even many gigabytes) of digital information, so that even non-specialists can easily understand them.

If all these requirements are satisfied, then the treatment of the mathematical models will lead to very large computational tasks. Indeed, the application of discretization and splitting procedures leads to several systems of ordinary differential equations. Every system may contain several millions of equations and has to be treated during many time-steps (as a rule several thousand time-steps). It is clear that these computational problems will cause difficulties even when big modern computers are used. This is why it is often necessary to perform some simplifications in the model. Such simplifications must be made so that the output results are still reliable. In order to satisfy the last requirement, one has to investigate how the changes of some parameters or some physical and chemical mechanisms will influence the output results. If the output results are not very sensitive to the variations of certain parameter or mechanism, then this means that the model can be simplified by choosing a simpler algorithm to calculate the parameter under consideration or to describe the mechanism under consideration by a simpler algorithm. If the output results are sensitive to changes in a given parameter of algorithm, then one must be more careful: the parameter must be calculated in a more accurate way (by a more complicated algorithm), the mechanism must be described by a more advanced (and, again, more complicated) algorithm. This short discussion shows that it is useful to perform some sensitivity analysis in order to understand better the relationships between parameters and/or mechanisms used in the model and the output results.

The chemical reactions that take place under the transport are one of the most important processes. Therefore, we decided to concentrate our attention to the chemical reactions. We shall not represent a comprehensive sensitivity analysis of the relationships between changes of the chemical rate constants and the output results, because this is a very difficult task. Instead of this we shall try to determine which reaction is the most important one for two selected chemical species (nitrogen di-oxide and ozone). Then we shall study the influence of changes of the chemical rate constant of this reaction on the nitrogen di-oxide and ozone concentrations. This is done by using a special Monte Carlo sensitivity technique.

4.1 The Danish Eulerian Model

The Danish Eulerian Model, see [Zl95] is described mathematically by the following system of partial differential equations:

$$\begin{aligned} \frac{\partial c_s}{\partial t} &= -\frac{\partial(uc_s)}{\partial x} - \frac{\partial(vc_s)}{\partial y} - \frac{\partial(wc_s)}{\partial z} \\ &+ \frac{\partial}{\partial x}\left(K_x \frac{\partial c_s}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial c_s}{\partial y}\right) + \frac{\partial}{\partial z}\left(K_z \frac{\partial c_s}{\partial z}\right) \\ &+ E_s + Q_s(c_1, c_2, \dots c_q) - (k_{1s} + k_{2s})c_s \ , \quad s = 1, 2, \dots q \ . \end{aligned} \tag{61}$$

The number q of equations in this system is equal to the number of chemical species that are studied by the model. This number varies from 10 to 168 in the experiments described in [Zl95]. The other quantities involved in the model can be described as follows (see again [Zl95] for more details):

- the unknowns c_s are concentrations of the chemical species,
- u, v and w are the components of the wind along the coordinate axes,
- K_x, K_y and K_z are diffusion coefficients,
- the emissions in the space domain are represented by the functions E_s,
- k_{1s} and k_{2s} are coefficients of dry and wet deposition respectively ($s = 1, \dots q$),
- the chemical reactions between species are described by the non-linear functions $Q_s(c_1, c_2, \dots c_q)$, where $s = 1, 2, \dots, q$ (the condensed CBM IV scheme that was

proposed by [GWKD89], see also [Zl95], is the particular chemical scheme which will be used in this paper).

It is very difficult to treat directly the system (1). Therefore, some kind of splitting is to be used. Splitting according to the major physical processes is very popular; see, for example, [Ma85], [MGS84] and [Zl95]. Such splitting procedures lead often to five sub-models which are to be treated cyclicly at every time-step ([Zl95]). These sub-models are describing the horizontal advection, the horizontal diffusion, the chemical reactions (64) including the emissions, the deposition and the vertical exchange.

$$\frac{\partial c_s^{(1)}}{\partial t} = -\frac{\partial (u c_s^{(1)})}{\partial x} - \frac{\partial (v c_s^{(1)})}{\partial y} \tag{62}$$

$$\frac{\partial c_s^{(2)}}{\partial t} = \frac{\partial}{\partial x}\left(K_x \frac{\partial c_s^{(2)}}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial c_s^{(2)}}{\partial y}\right) \tag{63}$$

$$\frac{\partial c_s^{(3)}}{\partial t} = E_s + Q_s(c_1^{(3)}, c_2^{(3)}, \ldots c_q^{(3)}) \tag{64}$$

$$\frac{\partial c_s^{(4)}}{\partial t} = -(k_{1s} + k_{2s}) c_s^{(4)} \tag{65}$$

$$\frac{\partial c_s^{(5)}}{\partial t} = -\frac{\partial (w c_s^{(5)})}{\partial z} + \frac{\partial}{\partial z}\left(K_z \frac{\partial c_s^{(5)}}{\partial z}\right) \tag{66}$$

Five large systems of ordinary differential equations can be obtained from the sub-models (62)-(66) by applying any space discretization method:

$$dg^{(i)}/dt = f^{(i)}(t, g^{(i)}) , \quad g^{(i)} \in \mathcal{R}^{N_x \times N_y \times N_z \times N_s} , \quad f^{(i)} \in \mathcal{R}^{N_x \times N_y \times N_z \times N_s} , \tag{67}$$

where N_x, N_y and N_z are the numbers of grid-points on the grid-lines parallel to the coordinate axes and $N_s = q$ is the number of chemical species involved in the model. The function $f^{(i)}$ is a vector-function whose components are approximations to the concentrations at the grid-points, while $f^{(i)}$ depends on the particular discretization method applied to the corresponding sub-model (see [Zl95]).

The number of equations in any of the five ODE systems in (67) is equal to the product of the number of grid-points and the number of species. Thus, if $N_x = 96$, $N_y = 96$, $N_z = 10$ and $N_s = 35$, then every ODE system contains 3225600 equations. Furthermore, the five ODE systems are to be treated numerically during many time-steps (typically several thousand time-steps are needed). This shows that the numerical treatment of large-scale air pollution models leads to very big computational problems. This is why both fast numerical algorithms and high speed computers are to be used.

The chemical sub-model is the most time-consuming part of the model. Therefore, one should be very careful when numerical algorithms for this sub-model are selected. The algorithms chosen must be fast. Furthermore, these must be efficient when high speed computers are used. The difficulties that have to be overcome when the chemical sub-model is treated are discussed in [SZ97], and we shall not repeat this discussion here. It is important, however, to emphasize the fact that the chemical sub-model consists of $N_x \times N_y \times N_z$ *independent* ODE systems. Each of these systems contains $N_s = q$ equations and can be rewritten as

$$dg/dt = f(t, g), \qquad g \in \mathcal{R}^{N_s}, \qquad f \in \mathcal{R}^{N_s}, \tag{68}$$

where g is a vector whose components are approximations to the concentrations at a given grid-point, while the right-hand-side vector f depends on the chemical mechanism which is used in the model. It is clear now that (68) can sometimes be considered (instead of the whole chemical sub-model) in studies of some phenomena which are directly connected to the chemical scheme. The much simpler model (68) will be called the box-model.

We shall use the two-dimensional version of the model.

4.2 An Algorithm for Monte Carlo Simulation

Monte Carlo (MC) simulation methods are methods for solving problems from different fields of science and engineering by using random variables. In general, the MC methods replace the task of finding an approximation to the exact solution with the task of finding an approximation of the mathematical expectation and some other statistical characteristics of the solution of the problem solved. Let the scalar variable J be the desired solution of the problem or some desired linear functional of the solution. A random variable ξ with a mathematical expectation equal to J must be constructed so that $E\xi = J$. An approximation to J can be computed as a mean value by using n independent values $\xi_1, \xi_2, \ldots, \xi_n$, of ξ,

$$J \approx \frac{1}{n} \left(\xi_1 + \xi_2 + \ldots + \xi_n \right). \tag{69}$$

Monte Carlo simulations are often used to evaluate some important statistical characteristics of the solution, such as the standard deviation, variance, skewness and, so on (see, for example, [So73] for more details)

Very often the Monte Carlo simulation is used to study complicated systems and phenomena when these can be treated as a *black box*. The main idea is to "introduce" some random parameter or random field in the input of the *black box* and to treat the calculated results by using well known statistical methods. There are two main problems in a such Monte Carlo simulation:

- how to simulate the input random parameter with a priori given statistical characteristics (mean value, standard deviation, and so on) in order to be sure that the considered realization of the random parameter is "near" to the theoretical one in some measured space

and

- how to handle the output results in order to measure the sensitivity of the *black box* to the considered input parameter?

There are many well known generators of *normally distributed* or *uniformly distributed* random variables, which can be used for producing high quality "input" data with a priori given statistical characteristics. The generator of the uniformly distributed random variables used by us is based on a special function $f(x)$, see [So73], which:

1. is defined on the interval $[0, 1]$,
2. satisfies $0 \leq f(x) \leq 1$,
3. its graph is "dense" in the unite cube.

It is necessary to take a linear combination of a large number k of values of the defined above special function $f(x)$ in order to obtain *normally distributed* random variables.

4.3 Finding the reaction that has greatest influence on the nitrogen di-oxide and ozone concentrations

The box model is defined by the system of ordinary differential equations (spdis1). This model has been used to find the reaction that has greatest influence on the concentrations of nitrogen di-oxide and ozone. The Monte Carlo algorithm sketched in the previous section is used in the experiments. In each experiment a sequence of 100 normally distributed random values of the chemical rate constant of one of the chemical reactions was produced by using a random number generator. After that the box model was run for these 100 random values. This procedure has been carried out for all 70 chemical reactions involved in the chemical scheme used in the Danish Eulerian Model; the condensed CBM IV scheme (see [GWKD89] and [Zl95]). This means that 100 runs were perform per each chemical reaction with normally distributed random values. The standard deviations of the nitrogen di-oxide and ozone concentrations produced when the chemical rate constants were varied as described above were compared. It has been found in this way that the most important chemical reaction for the nitrogen di-oxide and ozone concentrations is:

$$O_3 + NO \implies NO_2 \, . \tag{70}$$

This means that small changes of the rate constant of this reaction lead to considerably large changes in the concentrations of nitrogen di-oxide and ozone and, moreover, these changes are larger than the changes, for the same two chemical species, observed when the rate constants of the other chemical reactions were varied by using the same procedure.

4.4 Sensitivity tests with the Danish Eulerian Model

Consider the reaction (70). It is interesting to see what will happen if the experiment with a sequence of 100 normally distributed random values of the constant rate of reaction (70) is performed by using the two-dimensional version of the Danish Eulerian Model (instead of the box model used in the previous section). Several such experiments, each of them consisting of 100 runs with the Danish Eulerian Model, have been carried out. The standard deviation used in the calculating the sequence of normally distributed random values was varied in these experiments. Results obtained when the sequences of 100 normally distributed random values of the constant rate of reaction (70) are produced by using standard deviations $\sigma = 0.5$ and $\sigma = 0.25$ will be presented here, but some other values of the standard deviations were also used.

The results, obtained after performing 100 runs with normally distributed random values of the constant rate of reaction (70), were used to calculate the standard deviations and the skewness of the nitrogen dioxide and ozone concentrations for every value of the standard deviation σ used in the experiments. The results are presented in Figure 3 - Figure 8 (where by X and $Y = \sigma$ the mean value and the standard deviation of the randomly generated normalized rate constants are denoted). The following major conclusions can be drawn from this experiment:

- The standard deviations of the ozone concentrations given in Figure 3 for $\sigma = 0.5$ and Figure 5 for $\sigma = 0.25$ are greatest in the areas where the European emissions are biggest (compare Figure 3 and Figure 5 with Figure 1) and where the nitrogen di-oxide concentrations are highest (compare Figure 3 and Figure 5 with Figure 2).

- The patterns of the distributions of the standard deviations of the nitrogen dioxide concentrations are not so pronounced (see Figure 4 and Figure 6, where the results obtained, respectively, with $\sigma = 0.5$ and $\sigma = 0.25$ are given). Nevertheless, It is clear that the effect is opposite to the effect observed when the standard deviations of the ozone concentrations are studied. The standard deviations in the most polluted with nitrogen species areas are smaller than the standard deviation in the areas which are far away from the highly polluted areas; compare Figure 4 and Figure 6 with Figure 1 and Figure 2).

- If the standard deviation by which the sequences of normally distributed random values of the rate constant of reaction (70) is reduced, then the standard deviations of the ozone concentrations are also reduced, but the pattern of the distribution of the highest standard deviations remains the same; compare Figure 3 with Figure 5. The same is also true for the pattern of distribution of the highest standard deviations of the nitrogen di-oxide concentrations; compare Figure 4 with Figure 6.

- Results concerning the distribution of the skewness (for the sequence obtained with standard deviation $\sigma = 0.5$) are given in Figure 7 for the skewness of the ozone concentrations and in Figure 8 for the skewness of the nitrogen di-oxide concentrations. The two plots indicate that also here is the effect opposite: in the areas where the skewness of the ozone concentrations is greatest, the skewness of the nitrogen di-oxide concentrations is smallest.

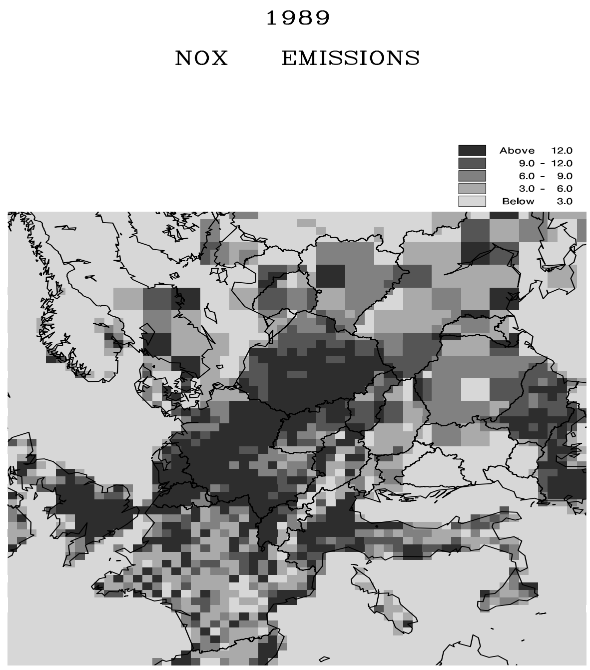

Figure 1
European nitrogen oxides emissions

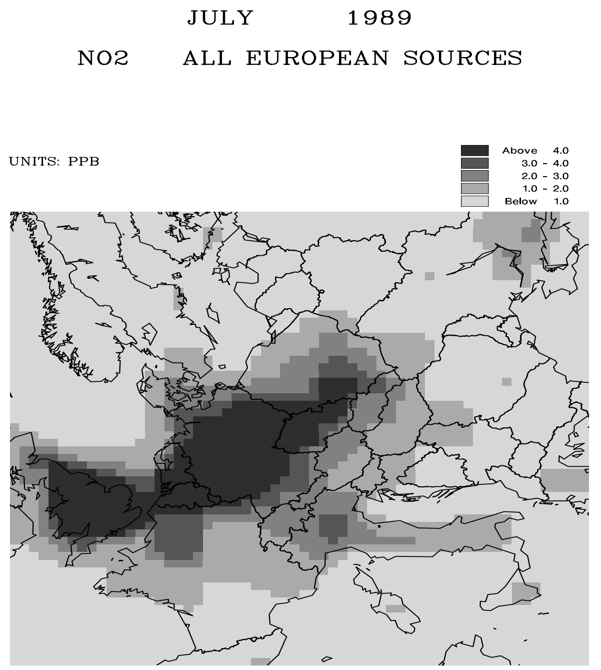

Figure 2
Nitrogen di-oxide concentrations in Europe

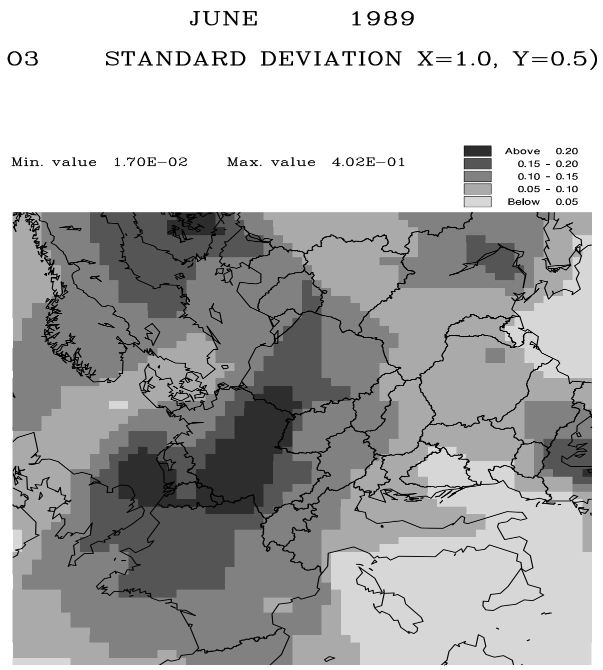

Figure 3
Standard deviations of the ozone concentrations (variance 0.50)

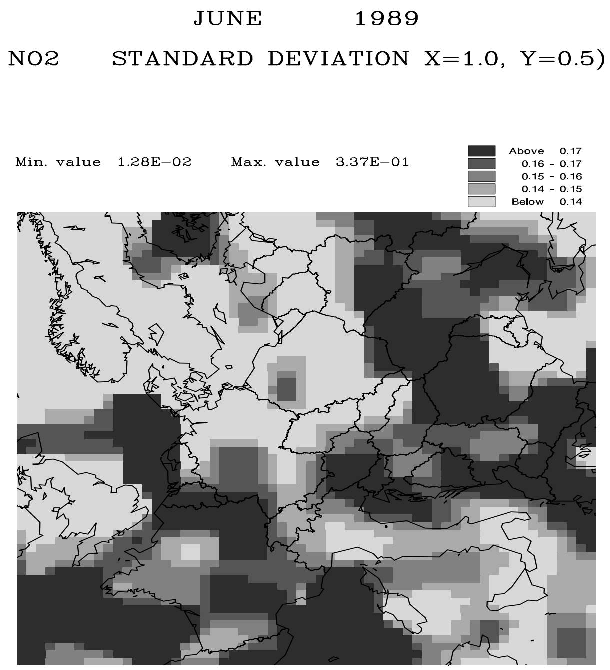

Figure 4
Standard deviations of the nitrogen di-oxide concentrations (variance 0.50)

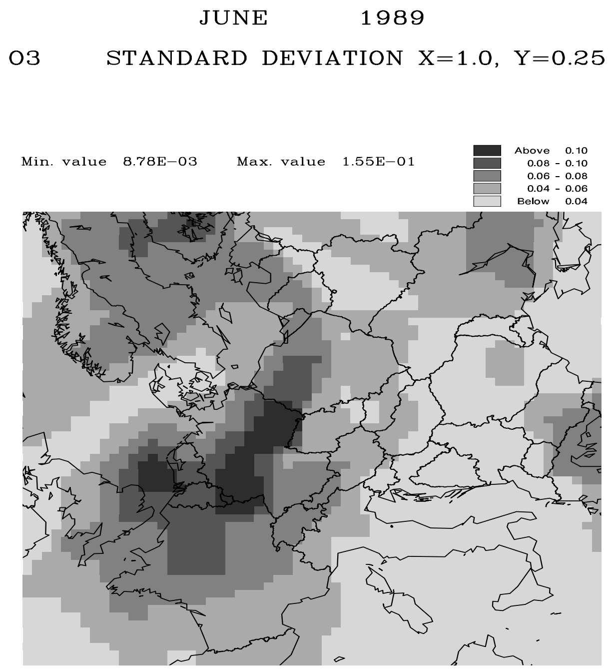

Figure 5
Standard deviations of the ozone concentrations (variance 0.25)

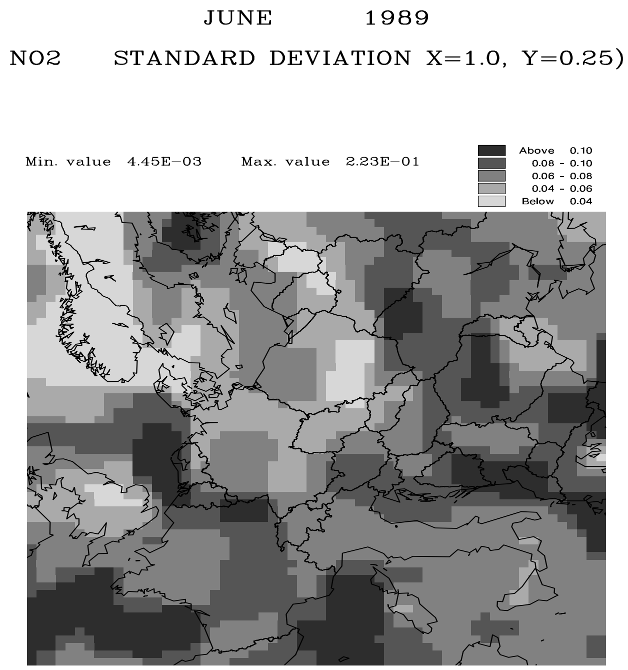

Figure 6
Standard deviations of the nitrogen di-oxide concentrations (variance 0.25)

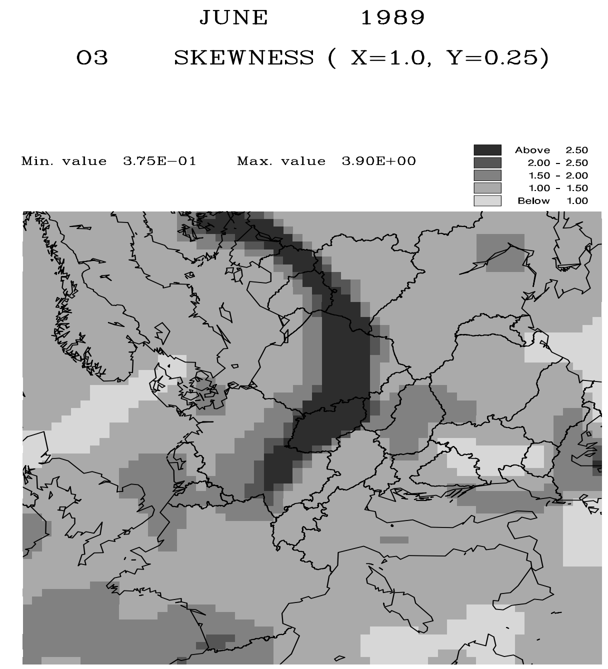

Figure 7
Skewness of the ozone concentrations (variance 0.50)

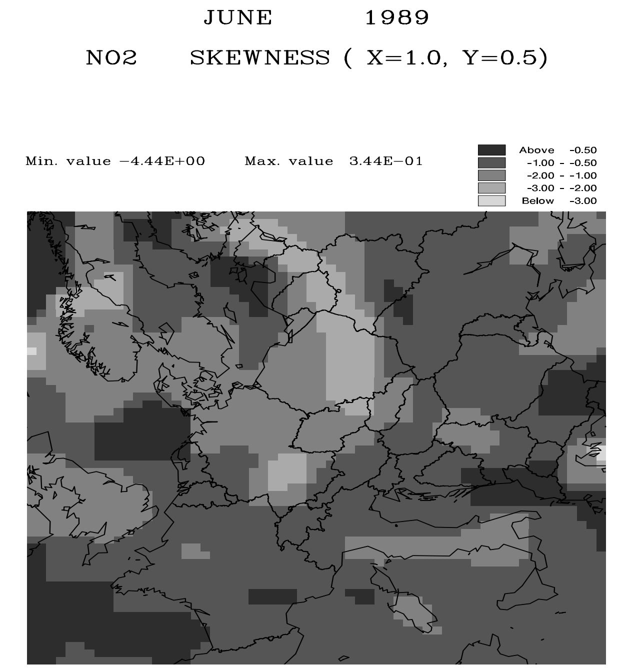

Figure 8
Skewness of the nitrogen di-oxide concentrations (variance 0.50)

The main result is that the influence of the rate constant of reaction (70) on the ozone concentrations seems to be great in the highly polluted with nitrogen pollutants areas. Therefore an accurate value of this rate constant is needed if the model is to be used on a space domain in which are highly polluted with nitrogen species areas. On the other hand, if in the space domain of the model there are not areas which are highly polluted with nitrogen species, then the accuracy with which this rate coefficient is determined becomes less important.

4.5 Concluding remarks

It is probably worthwhile to carry out some computations with the three-dimensional version of the Danish Eulerian Model (see [ZDG96]. The experiments in this paper have shown that different areas in the horizontal plane are influenced in a different way when certain constant rates are changed. The experiments with the three-dimensional version of the Danish Eulerian Model may show differences also in the vertical directions. However, such runs are very time-consuming. It will be possible to perform such experiments only if new and faster numerical methods are used in the model and, moreover, if bigger and faster computers become available.

It is desirable to use the Danish Eulerian Model in similar experiments, but for some other chemical rate constants. Maybe, one should also test the combined effect of changing simultaneously several chemical rate constants on relevant concentrations and/or depositions.

The sensitivity of the model results to variations of some other parameters (as, for example, the boundary conditions) should also be studied.

References

[Bi82] A.V. Bitzadze, *Equations of the Mathematical Physics*, **Nauka**, Moscow, 1982.

[Cu54] Curtiss J.H., *Monte Carlo methods for the iteration of linear operators.* **J. Math. Phys., Vol. 32**, No 4 (1954), pp. 209 –232.

[Cu56] Curtiss J.H. *A theoretical comparison of the efficiencies of two classical methods and a Monte Carlo method for computing one component of the solution of a set of linear algebraic equations.*, **Proc. Symposium on Monte Carlo Methods** , **John Wiley and Sons**, 1956, pp. 191–233.

[Di89] Dimov I.T., *Efficiency estimator for the Monte Carlo algorithms.* **Numerical Methods and Applications**, Proc. **II Intern. Conf. on Numerical Methods and Appl.** , Sofia, Publ. House of the Bulg. Acad. Sci., Sofia, 1989, pp. 111–115.

[Di91] Dimov I.T., *Minimization of the probable error forsome Monte Carlo methods* - Proc. of the Summer School on **Mathematical Modelling and Scientific Computations**, 23-28. 09. 1990, Albena, Bulgaria, Sofia, Publ. House of the Bulg. Acad. Sci., 1991, pp. 159–170.

[DG97] I.T. Dimov, T.V. Gurov, *Estimates of the computational complexity of iterative Monte Carlo algorithm based on Green's function approach*, in: **IMACS Seminar on Monte Carlo methods and Applications**, Brussels, April, 1997, to be published, 1997.

[DT93] I. Dimov, O. Tonev, *Random walk on distant mesh points Monte Carlo methods*, **J. of Statistical Physics, Vol. 70**(5/6), 1993, pp. 1333–1342.

[DT93a] 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.

[DK94] I. Dimov, A. Karaivanova, *Overconvergent Monte Carlo Methods for Density-function Modelling Using B-Splines*, **Adv. in Numerical Methods and Appl.**, **World Scientific**, 1994, pp. 85–93.

[DK94a] I. Dimov, A. Karaivanova, *Monte Carlo Parallel Algorithms*, Proc. III Intern. Conf. on **Applications of Supercomputers in Engineering - ASE/93, Comp. Mech. Publ., Elsevier Appl. Sci.**, London, New York, 1993, pp. 479–495.

[DKKS96] I. Dimov, A. Karaivanova, H. Kuchen, H. Stoltze, *Monte Carlo Algorithms for Elliptic Differential Equations. Data Parallel Functional Approach*, **Journal of Parallel Algorithms and Applications, Vol. 9** (1996), pp. 39–65.

[Di94] I Dimov, *Efficient and Overconvergent Monte Carlo Methods. Parallel algorithms.*, **Addvances in Parallel Algorithms**, (I. Dimov, O. Tonev, Eds.), Amsterdam, **IOS Press** (1994), pp. 100–111.

[DK96a] Ivan T. Dimov, Aneta N. Karaivanova, *A Fast Monte Karlo Method for Matrix Computations*, in **Iterative Methods in Linear Algebra II, IMACS Series in Computational and Applied Mathematics** (S. Margenov and P.S. Vassilevski Eds.), 1996, pp. 204–213.

[DK96] I.T.Dimov, A.N.Karaivanova, *Iterative Monte Carlo algorithms for linear algebra problems*, First Workshop on Numerical Analysis and Applications, Rousse, Bulgaria, June 24-27, 1996, in : **Numerical Analysis and Its Applications, Springer Lecture Notes in Computer Science, ser. 1196**, pp. 150–160.

[DJV96] I. Dimov, U. Jaekel, H. Vereecken, *A Numerical Approach for Determination of Sources in Transport Equations*, **J. Computers and Mathematics with Applications, Vol. 32, No. 5** (1996), pp. 31–42.

[Er84] Ermakov S.M., *On summation of series connected with integral equation.* **Vestnik Leningrad Univ. Math. Vol. 16** (1984) pp. 57–63.

[EM82] S.M. Ermakov, G.A. Mikhailov, *Statistical Modeling*, **Nauka**, Moscow, 1982.

[ENS84] S.M.Ermakov, V.V.Nekrutkin, A.S.Sipin, *Random processes for solving classical equations of mathematical physics*, **Nauka**, Moscow, 1984.

[GWKD89] M. W. Gery, G. Z. Whitten, J. P. Killus, M. C. Dodge, *A photochemical kinetics mechanism for urban and regional modeling*, **J. Geophys. Res., Vol. 94** (1989), pp. 12925–12956.

[GV83] G. H. Golub, C.F. Van Loon, *Matrix computations*, **The Johns Hopkins Univ. Press**, Baltimore, 1983.

[HH64] J.M. Hammersley, D.C. Handscomb, *Monte Carlo methods*, **John Wiley & Sons, inc.**, New York, London, Sydney, Methuen, 1964.

[KA77] L.V. Kantorovich, G.P. Akilov, *Functional analysis*, **Nauka**, Moscow, 1977.

[Ma85] G. I. Marchuk, *Mathematical modeling for the problem of the environment*, **Studies in Mathematics and Applications, No. 16**, North-Holland, Amsterdam, 1985.

[MGS84] G. J. McRae, W. R. Goodin, J. H. Seinfeld, *Numerical solution of the atmospheric diffusion equations for chemically reacting flows*, **J. Computational Physics, Vol. 45,** 1984, pp. 1–42.

[MAD94] G. Megson, V. Aleksandrov, I. Dimov, *Systolic Matrix Inversion Using a Monte Carlo Method*, **J. of Parallel Algorithms and Appl., Vol. 3, No 3/4** (1994), pp. 311–330.

[MU49] N. Metropolis, S. Ulam, *The Monte Carlo Method*, **J. of Amer. Statistical Assoc., 44**, (1949), No. 247, pp. 335–341.

[Mi83] Mikhailov V.P., *Partial differential equations,* Moskow, **Nauka**, 1983.

[Mi55] C. Miranda, *Equasioni alle dirivate parziali di tipo ellittico*, **Springer-Verlag**, Berlin, 1955.

[Sa89] K. Sabelfeld, *Algorithms of the Method Monte Carlo for Solving Boundary Value Problems*, **Nauka**, Moscow, 1989.

[Sh64] Shreider Yu.A., *Method of Statistical Testing. Monte Carlo method.* **Elsevierm Publishing Co.,** 335 Jan Van Galenstraat, P.O. Box 211, Amsterdam (Netherlands), 1964.

[Sh66] Shreider Yu.A., *The Monte Carlo method. Method of Statistical Testing.* **Pergamon Press Ltd.,** Ozford, London, Edinburg, New York, Paris, Frankfurt, 1966.

[SZ97] S. Skelboe, Z. Zlatev, *Exploiting the natural partitioning in the numerical solution of ODE systems arising in atmospheric chemistry*, In: **"Numerical Analysis and Its Applications"** (L. Vulkov, J. Wasniewski and P. Yalamov, Eds.), pp. 458–465, Springer, Berlin, 1997.

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

[Zl95] Z. Zlatev, *Computer treatment of large air pollution models*, **Kluwer Academic Publishers**, Dordrecht-Boston-London, 1995.

[ZDG96] Z. Zlatev, I. Dimov, K. Georgiev, **Three-dimensional version of the Danish Eulerian Model**, **Zeitschrift für Angewandte Mathematik und Mechanik**, **Vol. 76,** (1996) S4, pp. 473–476.