III. Iterative Monte Carlo Methods for Linear Equations

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. 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 k significant figures. In this latter case there are two errors - *systematic* and *stochastic*. The systematic error depends both on the number of iterations performed and the characteristic values of the iteration operator, 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 involving dense matrices.

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 k^{th} iteration. It is desired that

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

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. Sometimes the *iterative* *stationary linear Monte Carlo algorithms* are called *Power* Monte Carlo algorithms. The reason is that these algorithms find an approximation of a functional of *powers* of linear operators. In literature this class of algorithms is also known as Markov chain Monte Carlo since the statistical estimates can be considered as weights of Markov chains. We consider how such weights can be defined. We are focusing on two kind of linear operators: integral operators and matrices.

Iterative Monte Carlo Algorithms

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 $\mathbb{R}^d$ and $L = L(u)$ be a linear operator defined on $\mathbf{X}$.

Consider the sequence $u_1, u_2, ...,$ defined by the recursion formula

$$u_k = L(u_{k-1}) + f, \;\; k = 1, 2, \ldots \tag{39}$$

The formal solution of (39) is the truncated Neumann series

$$u_k = f + L(f) + \cdots + L^{k-1}(f) + L^k(u_0), \qquad k > 0, \tag{40}$$

where L^k means the k^{th} iterate of L.

As an example consider the integral iterations.

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

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

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,

$$L^2u(x) = \int_\Omega \int_\Omega l(x, x')l(x', x'')dx'dx''.$$

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

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

$$u = L(u) + f. \tag{41}$$

The truncation error of (40) 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

$$\mathbf{T}_{i+1} = \underbrace{\mathbb{R}^d \times \mathbb{R}^d \times \cdots \times \mathbb{R}^d}_{i \text{ times}}, \qquad i = 1, 2, \ldots, k, \tag{42}$$

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

$$J(u_k) \approx \frac{1}{N} \sum_{s=1}^{N} \{\theta_k\}_s, \tag{43}$$

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

The probable error r_N of (43) is then

$$r_N = c\ \sigma(\theta_k) N^{-\frac{1}{2}},$$

where $c \approx 0.6745$ and $\sigma(\theta_k)$ 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

$$L(u) = \int_\Omega l(x,y)u(y)dy$$

  and $u(x)$ and $f(x)$ are functions.

First consider the second case. Equation (41) becomes

$$u(x) = \int_\Omega l(x,y)u(y)dy + f(x) \ \text{ or } \ u = Lu + f. \tag{44}$$

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

$$J(u) = \int_{\Omega} h(x)u(x)dx = (u, h). \tag{45}$$

In fact, equation (45) defines an inner product of a given function $h(x) \in \mathbf{X}$ with the solution of the integral equation (41).

Sometimes, the adjoint equation

$$v = L^* v + h \tag{46}$$

will be used.

In (46) $v, h \in \mathbf{X}^*$, $L^* \in [\mathbf{X}^* \to \mathbf{X}^*]$, where $\mathbf{X}^*$ is the dual functional space to $\mathbf{X}$ and L^* is an adjoint operator.

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

$$\| f \|_{\mathbf{L}_1} = \int\limits_{\Omega} | f(x) | \, dx;$$

$$\| L \|_{\mathbf{L}_1} \leq \sup_x \int\limits_{\Omega} | l(x, x') | \, dx'. \tag{47}$$

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

$$\| h \|_{\mathbf{L}_\infty} = \sup |h(x)| \;\; x \in \Omega.$$

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 (45) is finite. In fact,

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

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

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

Let us integrate the last inequality with respect to x:

$$\int_\Omega |Lu|^2 dx \le \int_\Omega \int_\Omega l^2(x, x') dx' dx \int_\Omega u^2(x') dx' < \infty.$$

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

Obviously, if $u \in \mathbf{L}_1$ and $h \in \mathbf{L}_\infty$ the inner product (45) 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 was shown by K. Sabelfeld, it is possible to consider a Monte Carlo algorithm 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 algorithm when the Neumann series for the original problem does not converge or to accelerate the convergence when it converges slowly.

It is easy to show that

$$J = (h, u) = (f, v).$$

In fact, let us multiply (44) by v and (46) by u and integrate. We obtain

$$(v, u) = (v, Lu) + (v, f) \text{ and } (v, u) = (L^* v, u) + (h, u).$$

Since

$$(L^* v, u) = \int_\Omega L^* v(x) u(x) dx = \int_\Omega \int_\Omega l^*(x, x') v(x') u(x) dx dx'$$

$$= \int_\Omega \int_\Omega l(x', x)u(x)v(x')dxdx' = \int_\Omega Lu(x')v(x')dx' = (v, Lu),$$

we have

$$(L^*v, u) = (v, Lu).$$

Thus, $(h, u) = (f, v)$. Usually, the kernel $l(x', x)$ is called *transposed kernel*.

Consider the Monte Carlo algorithm for evaluating the functional (45). It can be seen that when $l(x, x') \equiv 0$ evaluation of the integrals can pose a problem. Consider a random point $\xi \in \Omega$ with a density $p(x)$ and let there be N realizations of the random point ξ_i $(i = 1, 2, \ldots, N)$. Let a random variable $\theta(\xi)$ be defined in Ω, 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 $Markov\ chains$ (see, Definition **??** given in Introduction). Markov chain Monte Carlo is also known as the Metropolis or the Metropolis-Hastings method. Here this method is considered as a special class of iterative stochastic methods, since such a consideration helps to obtain some error analysis results.

An approximate value of the linear functional J, defined by (45), 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

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

where $W_0 = 1$; $W_j = W_{j-1}\dfrac{l(\xi_{j-1},\xi_j)}{p(\xi_{j-1},\xi_j)}, j = 1, 2, \ldots,$ and $\xi_0, \xi_1, \ldots$ is a Markov chain in Ω 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 (40) can be written in the following form:

$$u_k = L^k u_0 + L^{k-1} f + \cdots + Lf + f = (I - L^k)(I - L)^{-1} f + L^k u_0, \qquad (48)$$

where I is the unit (identity) matrix; $L = (l_{ij})_{i,j=1}^{n}$; $u_0 = (u_1^0, \ldots, u_n^0)^T$ 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 then there exists a vector u such that

$$u = \lim_{k \to \infty} u_k \ ,$$

which satisfies the equation

$$u = Lu + f. \qquad (49)$$

Now consider the problem of evaluating the inner product

$$J(u) = (h, u) = \sum_{i=1}^{n} h_i u_i \, , \tag{50}$$

where $h \in \mathbb{R}^{n \times 1}$ is a given vector-column.

To define a random variable whose mathematical expectation coincides with the functional (50) for the system (52) first consider the integral equation (44) for which $\Omega = [0, n)$ is an one-dimensional interval divided into equal subintervals $\Omega_i = [i-1, i)$, $i = 1, 2, \dots n$ such that

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

Then the integral equation (44) becomes

$$u_i = \sum_j \int_{\Omega_j} l_{ij} u(y) dy + f_i$$

for $u_i \in \Omega_i$. Denote

$$u_j = \int_{\Omega_j} u(y) dy \tag{51}$$

so that one obtains, for $u(x) \in \Omega_i$,

$$u(x) = \sum_{j=1}^{n} l_{ij} u_j + f_i.$$

From the last equation it follows that $u(x) = u_i$ and so,

$$u_i = \sum_{j=1}^{n} l_{ij} u_j + f_i \ ,$$

or in a matrix form

$$u = Lu + f \ , \tag{52}$$

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

The above presentation permits the consideration of the following random variable

$$\theta[h] = \frac{h_{\alpha_0}}{p_0} \sum_{\nu=0}^{\infty} W_\nu f_{\alpha_\nu} \ , \tag{53}$$

where

$$W_0 = 1; \qquad W_\nu = W_{\nu-1} \frac{l_{\alpha_{\nu-1},\alpha_\nu}}{p_{\alpha_{\nu-1},\alpha_\nu}}, \qquad \nu = 1, 2, \ldots \tag{54}$$

and $\alpha_0, \alpha_1, \ldots$ is a Markov chain on elements of the matrix L created by using an initial probability p_0 and a transition probability $p_{\alpha_{\nu-1},\alpha_\nu}$ for choosing the element $l_{\alpha_{\nu-1},\alpha_\nu}$ of the matrix L.

Solving Linear Systems and Matrix Inversion

Consider a matrix L:

$$L = \{l_{ij}\}_{i,j=1}^{n}, \quad L \in \mathbb{R}^{n \times n}$$

and a vector

$$f = (f_1, \ldots, f_n)^T \in \mathbb{R}^{n \times 1}$$

The matrix L can be considered as a linear operator $L[\mathbb{R}^n \to \mathbb{R}^n]$, so that the linear transformation

$$Lf \in \mathbb{R}^{n \times 1} \tag{55}$$

defines a new vector in $\mathbb{R}^{n \times 1}$.

Since iterative Monte Carlo algorithms using the transformation (55) will be considered, the linear transformation (55) will be called an *iteration*. The algebraic transformation (55) 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

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

of the solution $u \in \mathbb{R}^{n\times 1}$ of the linear algebraic system

$$Au = b,$$

where $A = \{a_{ij}\}_{i,j=1}^{n} \in \mathbb{R}^{n\times n}$ is a given matrix; $b = (b_1, \dots, b_n)^T \in \mathbb{R}^{n\times 1}$ and $h = (h_1, \dots, h_n)^T \in \mathbb{R}^{n\times 1}$ are given vectors.

It is possible to choose a non-singular matrix $M \in \mathbb{R}^{n\times n}$ such that $MA = I - L$, where $I \in \mathbb{R}^{n\times n}$ is the identity matrix and $Mb = f$, $f \in \mathbb{R}^{n\times 1}$.

Then

$$u = Lu + f.$$

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. If the conditions (i) are fulfilled, then (53), (54) become a *stationary linear iterative Monte Carlo algorithm*.

As a result the convergence of the Monte Carlo algorithm depends on truncation error of (48).

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

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

where $A \in \mathbb{R}^{n\times n}$ is a given real matrix. The matrix inversion is not often used in practical computations since this operation is computationally expensive. Nevertheless, for some problem (like approximate finding of good matrix preconditioners) algorithms for matrix inversion with relatively low accuracy could be very important as a part of some numerical solvers.

Assume that the following conditions are fulfilled:

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

Obviously, if the condition (i) is fulfilled, the solution of **the problem P1** can be obtained using the iterations (48).

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

$$L = I - A$$

can be considered.

Since it is assumed that the conditions (ii) are fulfilled, the inverse matrix $C = A^{-1}$ can be presented as

$$C = \sum_{i=0}^{\infty} L^i.$$

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 $p = \{p_i\}_{i=1}^n \in \mathbb{R}^n$, such that $p_i \geq 0, i = 1, \ldots, n$ and $\sum_{i=1}^n p_i = 1$.

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

Define sets of *permissible* densities $\mathcal{P}_h$ and $\mathcal{P}_L$.

Definition 4. *The initial density vector $p = \{p_i\}_{i=1}^n$ is called permissible to* *the vector* $h = \{h_i\}_{i=1}^n \in \mathbb{R}^n$, *i.e.* $p \in \mathcal{P}_h$, *if*

$$p_i > 0, \ \textit{when} \ h_i \neq 0 \ \textit{ and } \ p_i = 0, \ \textit{ when } \ h_i = 0 \ \textit{ for } \ i = 1, \dots, n.$$

The transition density matrix $P = \{p_{ij}\}_{i,j=1}^n$ *is called* $permissible$ *to the matrix* $L = \{l_{ij}\}_{i,j=1}^n$, *i.e.* $P \in \mathcal{P}_L$, *if*

$$p_{ij} > 0, \textit{when} \ \ l_{ij} \neq 0 \ \textit{ and } \ p_{ij} = 0, \ \textit{ when } \ l_{ij} = 0 \ \textit{ for } i, j = 1, \dots, m.$$

Note that the set of $permissible$ densities is a subset of $tolerant$ densities, defined with respect to density functions.

Consider the following Markov chain:

$$T_i = \alpha_0 \rightarrow \alpha_1 \rightarrow \dots \rightarrow \alpha_i, \tag{56}$$

where $\alpha_j = 1, 2, \dots, i$ for $j = 1, \dots, i$ are natural random numbers.

The rules for defining the chain (56) are:

$$Pr(\alpha_0 = \alpha) = p_\alpha, \quad Pr(\alpha_j = \beta|\alpha_{j-1} = \alpha) = p_{\alpha\beta}. \tag{57}$$

Assume that

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

Now define the random variables W_ν using the formula (54). One can see, that the random variables W_ν, where $\nu = 1, \dots, i$, can also be considered as weights on the Markov chain (57).

From all possible *permissible* densities we choose the following

$$p = \{p_\alpha\}_{\alpha=1}^{n} \in \mathcal{P}_h, \quad p_\alpha = \frac{|h_\alpha|}{\sum_{\alpha=1}^{n} |h_\alpha|}; \tag{58}$$

$$P = \{p_{\alpha\beta}\}_{\alpha,\beta=1}^{n} \in \mathcal{P}_L, \quad p_{\alpha\beta} = \frac{|l_{\alpha\beta}|}{\sum_{\beta=1}^{n} |l_{\alpha\beta}|}, \alpha = 1, \dots, n. \tag{59}$$

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 $p = \{p_\alpha\}_{\alpha=1}^{n}$ is called *almost optimal initial density vector* and the transition density matrix $P = \{p_{\alpha\beta}\}_{\alpha,\beta=1}^{n}$ is called *almost optimal density matrix* (Dimov). Such density distributions lead to almost optimal algorithms in the sense that for a class of matrices A and vectors h such a choice coincides with optimal weighted algorithms. The reason to use MAO instead of Uniform Monte Carlo is that MAO normally gives much smaller variances. On the other hand, the truly optimal weighted algorithms are very time consuming, since to define the optimal densities one needs to solve an additional integral equation with a quadratic kernel. This procedure makes the optimal algorithms very expensive.

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

$$\Delta_q = \delta_0 \times \delta_1 \times \cdots \times \delta_q .$$

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:

$$E\left\{W_i f_{\alpha_i}\right\} = (h, L^i f), \quad i = 1, 2, \dots ;$$

$$E\left\{\sum_{i=0}^{n} W_i f_{\alpha_i}\right\} = (h, u), \quad (P1),$$

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

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

Convergence and Mapping

We consider Monte Carlo algorithms for solving linear systems of equations and matrix inversion in the case when the corresponding Neumann series does not converge, or converges slowly.

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

$$u - \lambda L u = f, \tag{60}$$

where λ is some parameter. Note that the matrices can be considered as linear operators. Define resolvent operator (matrix) 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 (60), 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

$$R_\lambda = L + \lambda L^2 + \dots \, , \tag{61}$$

The systematic error of (61), when m terms are used, is

$$r_s = O[(|\lambda|/|\lambda_1|)^{m+1} m^{\rho-1}], \tag{62}$$

where ρ is the multiplicity of the root λ_1.

From (62) is follows that when λ is approximately equal to λ_1 the sequence (61) and the corresponding Monte Carlo algorithm converges slowly. When $\lambda \geq \lambda_1$ the algorithm does not converge.

Obviously, the representation (61) 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 defined when the sequence (61) does not converge.

To answer these questions we apply a $mapping$ of the spectral parameter λ in (60).

The algorithm under consideration follows an approach which is similar to the resolvent analytical continuation method used in functional analysis (Kantorovich, Akilov) and in integral equations.

To extend these results it is necessary to show that the mapping approach can be applied for any linear operators (including matrices). Consider the problem of constructing the solution of (60) for $\lambda \in \Omega$ and $\lambda \neq \lambda_k, k = 1, 2, \ldots$, where the domain Ω is a domain lying inside the definition domain of the $R_\lambda f$, such that all eigenvalues are outside of the domain Ω. In the neighborhood of the point $\lambda = 0$ $(\lambda = 0 \in \Omega)$ 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

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

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

$$R_{\psi(\alpha)} f = \sum_{j=0}^{\infty} b_j \alpha^j \ , \tag{63}$$

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 Ω 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 (63) converges; hence the solution of the functional equation (60) 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 (61) converges slowly.

To apply this approach one needs some information about the spectrum of the linear operator (respectively, the matrix. Let 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 ($\lambda = \lambda_* = 1$):

$$\lambda = \psi(\alpha) = \frac{4a\alpha}{(1-\alpha)^2}. \tag{64}$$

The sequence $R_{\psi(\alpha)} f$ for the mapping (64) converges absolutely and uniformly (Kantorovich, Akilov).

In Monte Carlo calculations we cut the sequence in (63) after m terms

$$R_{\lambda_*} f \approx \sum_{k=1}^{m} b_k \alpha_k^k = \sum_{k=1}^{m} \alpha_*^k \sum_{i=1}^{k} d_i^{(k)} c_i = \sum_{k=1}^{m} g_k^{(m)} c_k, \qquad (65)$$

where

$$g_k^{(m)} = \sum_{j=k}^{m} d_k^{(j)} \alpha_*^j. \qquad (66)$$

The coefficients

$$d_k^{(j)} = (4a)^k q_{k,j}$$

and $g_k^{(m)}$ can be calculated in advance.

The coefficients $d_k^{(j)}$ for the mapping (65) are calculated and presented in Table 5 (for $k, j \leq 9$) .

Table 5: Table of the coefficients $q_{k,j} = (4a)^{-k} d_k^{(j)}$ for $k, j \leq 9$.

k/j	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2		1	4	10	20	35	42	84	120
3			1	6	21	56	126	252	462
4				1	8	36	120	330	792
5					1	10	55	220	715
6						1	12	78	364
7							1	14	105
8								1	16
9									1

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 algorithm has to be used.

The mapping (64) creates the following Monte Carlo iteration process

$$\begin{aligned}
u_0 &= f, \\
u_1 &= 4aLu_0, \\
u_2 &= 4aLu_1 + 2u_1, \\
u_3 &= 4aLu_2 + 2u_2 - u_1, \\
u_j &= 4aLu_{j-1} + 2u_{j-1} - u_{j-2}, \;\; j \geq 3.
\end{aligned} \tag{67}$$

and from (67) we have

$$u^{(k)} = 4a\alpha Lu^{(k-1)} + 2\alpha u^{(k-1)} - \alpha^2 u^{(k-2)} + f(1-\alpha^2), \;\; k \geq 3.$$

A Highly Convergent Algorithm for Systems of Linear Algebraic Equations

Suppose we have a Markov chain with i states. The random trajectory (chain) T_i, of length i, starting in the state α_0 was defined as follows

$$T_i = \alpha_0 \to \alpha_1 \to \cdots \to \alpha_j \to \cdots \to \alpha_i,$$

where α_j means the number of the state chosen, for $j = 1, 2, \ldots, i$.

Assume that

$$P(\alpha_0 = \alpha) = p_\alpha, \quad P(\alpha_j = \beta | \alpha_{j-1} = \alpha) = p_{\alpha\beta},$$

where p_α is the probability that the chain starts in state α and $p_{\alpha\beta}$ is the transition probability to state β after being in state α. Probabilities $p_{\alpha\beta}$ define a transition matrix $P \in \mathbb{R}^{n\times n}$. We require that

$$\sum_{\alpha=1}^{n} p_\alpha = 1, \quad \sum_{\beta=1}^{n} p_{\alpha\beta} = 1, \quad \text{for any} \quad \alpha = 1, 2, \ldots, n.$$

Suppose the distributions created from the density probabilities p_α and $p_{\alpha\beta}$ are $permissible$, i.e. $p \in \mathcal{P}_h$ and $P \in \mathcal{P}_L$.

Now consider the problem of evaluating the inner product (50) $J(u) = (h, u) = \sum_{\alpha=1}^{n} h_\alpha u_\alpha$ of a given vector h with the vector solution of the system (52).

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

$$\theta_m^*[h] = \frac{h_{\alpha_0}}{p_0} \sum_{\nu=0}^{m} g_\nu^{(m)} W_\nu f_{\alpha_\nu}, \tag{68}$$

where $W_0 = 1$, $g_0^{(m)} = 1$ and

$$W_\nu = W_{\nu-1} \frac{l_{\alpha_{\nu-1},\alpha_\nu}}{p_{\alpha_{\nu-1},\alpha_\nu}}, \qquad \nu = 1, 2, \dots,$$

($\alpha_0, \alpha_1, \alpha_2, \dots$ is a Markov chain with initial density function p_{α_0} and transition density function $p_{\alpha_{\nu-1},\alpha_\nu}$) and coefficients $g_j^{(m)}$ are defined by (66) for $j \geq 1$.

The following theorem holds:

Theorem 13. *Consider matrix L, whose Neumann series (61) does not necessarily converge. Let (64) be the required mapping, so that the presentation (65) exists. Then*

$$E\left\{ \lim_{m\to\infty} \frac{h_{\alpha_0}}{p_0} \sum_{\nu=0}^{m} g_\nu^{(m)} W_\nu f_{\alpha_\nu} \right\} = (h, u).$$

Proof 8. *First consider the density of the Markov chain $\alpha_0 \to \alpha_1 \to \dots \to \alpha_i$*

as a point in $m(i+1)$-dimensional Euclidian space $\mathbf{T}_{i+1} = \underbrace{\mathbb{R}^n \times \ldots \times \mathbb{R}^n}_{i+1}$:

$$P\{\alpha_0 = t_0, \alpha_1 = t_1, \ldots, \alpha_i = t_i\} = p_0 p_{t_0 t_1} p_{t_1 t_2} \ldots p_{t_{i-1} t_i}.$$

Now calculate the mathematical expectation of the random variable

$$\frac{h_{\alpha_0}}{p_0} g_\nu^{(m)} W_\nu f_{\alpha_\nu}.$$

From the definition of the mathematical expectation it follows that:

$$\begin{aligned} E\left\{\frac{h_{\alpha_0}}{p_0} g_\nu^{(m)} W_\nu f_{k\nu}\right\} &= \sum_{t_0,\ldots,t_\nu=1}^{m} \frac{h_{t_0}}{p_0} g_\nu^{(m)} W_\nu f_{t_\nu} p_0 p_{t_0 t_1} \ldots p_{t_{\nu-1} t_\nu} \\ &= \sum_{t_0,\ldots,t_\nu=1}^{m} h_{t_0} l_{t_0 t_1} l_{t_1 t_2} \ldots l_{t_\nu - 1} t_\nu f_{t\nu} = (h, L^\nu f). \end{aligned}$$

The existence and convergence of the sequence (66) ensures the following representations:

$$\sum_{\nu=0}^{m} E\left|\frac{h_{\alpha_0}}{p_0} g_\nu^{(m)} W_\nu f_{k\nu}\right| = \sum_{\nu=0}^{m} (|h|, |L|^\nu |f|) = \left(|h|, \sum_{\nu=0}^{m} |L|^\nu |f|\right),$$

$$E\left\{\lim_{m\to\infty} \frac{h_{\alpha_0}}{p_0} \sum_{\nu=0}^{m} g_\nu^{(m)} W_\nu f_{\alpha_\nu}\right\}$$

$$= \quad \sum_{\nu=0}^{\infty} E\left\{\frac{h_{\alpha_0}}{p_0} g_\nu^{(m)} W_\nu f_{\alpha_\nu}\right\} = \sum_{\nu=0}^{\infty} (h, L^\nu f) = (h, u).$$

This theorem permits the use of the random variable $\theta_m^*[h]$ for calculating the inner product (50).

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

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

where the one is in the "r^{th}" position and "T" means transposition. It follows that

$$(h, u) = \sum_{\alpha}^{n} e_\alpha(r) u_\alpha = u_r$$

and the corresponding Monte Carlo algorithm is given by

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

where N is the number of chains and

$$\theta_m^*[e(r)]_s = \sum_{\nu=0}^{m} g_\nu^{(m)} W_\nu f_{\alpha_\nu};$$

$$W_\nu = \frac{l_{r\alpha_1} l_{\alpha_1\alpha_2} \dots l_{\alpha_{\nu-1}\alpha_\nu}}{p_{r\alpha_1} p_{\alpha_1\alpha_2} \dots p_{\alpha_{\nu-1}p_\nu}}.$$

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

$$L = I - A, \tag{69}$$

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 (60) be the following unit vector

$$f_{r'} = e(r').$$

Theorem 14. *Consider matrix L, whose Neumann series (61) does not necessarily converge. Let (64) be the required mapping, so that representation*

(65) exists. Then

$$E\left\{\lim_{m\to\infty}\sum_{\nu=0}^{m} g_{\nu}^{(m)} \frac{l_{r\alpha_1} l_{\alpha_1\alpha_2}\dots l_{\alpha_{\nu-1}\alpha_\nu}}{p_{r\alpha_1} p_{\alpha_1\alpha_2}\dots p_{\alpha_{\nu-1}} p_\nu} f_{r'}\right\} = c_{rr'}.$$

Proof 9. *The proof is similar to the proof of Theorem 13, but in this case we need to consider an unit vector $e(r)$ instead of vector h and vector $e(r')$ instead of $f_{k\nu}$:*

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

So, in this case the "r^{th}" component of the solution is estimated:

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

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

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

that is:

$$\begin{aligned}
& E \quad \left\{ \lim_{m\to\infty} \sum_{\nu=0}^{m} g_\nu^{(m)} \frac{l_{r\alpha_1} l_{\alpha_1\alpha_2} \ldots l_{\alpha_{\nu-1}\alpha_\nu}}{p_{r\alpha_1} p_{\alpha_1\alpha_2} \ldots p_{\alpha_{\nu-1}\alpha_\nu}} e(r') \right\} \\
= & \quad \lim_{m\to\infty} \sum_{\nu=0}^{\infty} E \left\{ g_\nu^{(m)} \frac{l_{r\alpha_1} l_{\alpha_1\alpha_2} \ldots l_{\alpha_{\nu-1}\alpha_\nu}}{p_{r\alpha_1} p_{\alpha_1\alpha_2} \ldots p_{\alpha_{\nu-1}\alpha_\nu}} e(r') \right\} = \sum_{\nu=0}^{\infty} (e(r), L^\nu e(r')) \\
= & \quad \left(e(r), \sum_{\nu=0}^{\infty} L^\nu e(r') \right) = \sum_{i=1}^{n} c_{ri} e(r') = c_{rr'}.
\end{aligned}$$

Theorem 14 permits the use of the following Monte Carlo algorithm for calculating elements of the inverse matrix C:

$$c_{rr'} \approx \frac{1}{N} \sum_{s=1}^{N} \left[\sum_{(\nu|\alpha_\nu = r')}^{m} g_\nu^{(m)} \frac{l_{r\alpha_1} l_{\alpha_1\alpha_2} \dots l_{\alpha_{\nu-1}\alpha_\nu}}{p_{r\alpha_1} p_{\alpha_1\alpha_2} \dots p_{\alpha_{\nu-1}p_\nu}} \right]_s ,$$

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

$$W_\nu^{(m)} = g_\nu^{(m)} \frac{l_{r\alpha_1} l_{\alpha_1\alpha_2} \dots l_{\alpha_{\nu-1}\alpha_\nu}}{p_{r\alpha_1} p_{\alpha_1\alpha_2} \dots p_{\alpha_{\nu-1}p_\nu}}$$

for which $\alpha_\nu = r'$ are included in the sum (69).

Observe that since $W_\nu^{(m)}$ is only contained in the corresponding sum for $r' = 1, 2, \dots, n$ 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 exploit later.

Balancing of 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 r_N (the probable error) must be approximately equal to the systematic one r_s, 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 m, i.e., $T = E(m)$.

Let us consider the case when the algorithm is applied to **Problem 1**. Using the mapping procedure and a random variable, defined by (68), we accelerate the convergence of the algorithm proposed in Curtiss, J.H. This means that for a fixed number of steps m

$$r_s(m) < r_s^{(C)}(m), \qquad (70)$$

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

$$\sigma(\theta^*) < \sigma(\theta) \tag{71}$$

and thus

$$r_N(\sigma(\theta^*)) < r_N^{(C)}(\sigma(\theta)), \tag{72}$$

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

Next consider the general error

$$R = r_N(\sigma) + r_s(m)$$

for matrix inversion by our Monte Carlo approach. Let R be fixed. Obviously from (70) and (71) it follows that there exist constants $c_s > 1$ and $c_N > 1$, such that

$$r_s^{(C)}(m) = c_s r_s,$$

$$r_N^{(C)}(\sigma) = c_N r_N.$$

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

$$R = R^{(C)} = r_N^{(C)}(\sigma) + r_s^{(C)}(m) = c_N r_N(\sigma) + c_s r_s(m).$$

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

$$\frac{c\sigma(\theta)}{N_c^{1/2}} + r_s^{(C)}(m) = \frac{c c_N \sigma(\theta^*)}{N_c^{1/2}} + c_s r_s(m)$$

$$= \frac{c\sigma(\theta^*)}{N^{1/2}} + r_s(m),$$

or

$$\frac{c\sigma(\theta^*)}{N^{1/2}} = \frac{c c_N \sigma(\theta^*)}{N_c^{1/2}} + (c_s - 1) r_s(m)$$

and

$$\frac{1}{N^{1/2}} = \frac{c_N}{N_c^{1/2}} + \frac{(c_s - 1) r_s(m)}{c\sigma(\theta^*)}, \tag{73}$$

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

Denote by b the following strictly positive variable

$$b = \frac{(c_s - 1) r_s(m)}{c\sigma(\theta^*)} > 0. \tag{74}$$

From (73) and (74) we obtain:

$$N = \frac{N_c}{\left(c_N + b N_c^{1/2}\right)^2}. \tag{75}$$

The result (75) 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 (74) 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 realizations of the Markov chain 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 Monte Carlo iterative process.

Estimators

Some estimates of N and the mathematical expectation for the length of the Markov chains T for Monte Carlo matrix inversion will now be outlined.

Using an almost optimal frequency function and according to the principle of collinearity of norms (Dimov, I., 1991) $p_{\alpha\beta}$ is chosen proportional to the $|l_{\alpha\beta}|$ (see, (59)). Depending on estimates of the convergence of the Neumann series one of the following stopping rules can be selected to terminate Markov chains:

- **(i)** when $|W_\nu^{(m)}| < \delta$;
- **(ii)** when a chain enters an absorbing state.

In the case of a Monte Carlo algorithm without any absorbing states ($f_{\alpha_j} = \delta_{\alpha_j\beta}$ if $\alpha\beta^{th}$ entry of inverse matrix can be computed) the bounds of T and $D\theta^*$ are

$$T \leq \frac{|\log \delta|}{|\log \|L\||}$$

and

$$D\theta^* \leq \frac{1}{(1-\|L\|)^2},$$

where the matrix norm $\|L\|$ is defined as $\|L\| = \|L\|_1 = \max_j \sum_{i=1}^n |l_{ij}|$. Consider the Monte Carlo algorithms with absorbing states where $\hat{\theta}[h]$ denotes a random variable $\hat{\theta}_T[h]$ (T is the length of the chain when absorption takes place) taken over an infinitely long Markov chain.

The bounds on T and $D\hat{\theta}[h]$ if the chain starts in state $r = \alpha$ and $p_{\alpha\beta} = |l_{\alpha\beta}|$, for $\alpha, \beta = 1, 2, ..., n$ are

$$E(T|r = \alpha) \leq \frac{1}{(1-\|L\|)},$$

and

$$D\hat{\theta}[h] \leq \frac{1}{(1-\|L\|)^2}.$$

According to the error estimation

$$N \geq \frac{0.6745^2}{\varepsilon^2} D\hat{\theta}[h]$$

for a given error ε. Thus

$$N \geq \frac{0.6745^2}{\varepsilon^2} \frac{1}{(1 - \|L\|)^2}$$

is a lower bound on N.

If low precision solutions (e.g. $10^{-2} < \varepsilon < 1$) are accepted it is clear that $N >> n$ as $N \mapsto \infty$. Consider N and T as functions of

$$\frac{1}{(1 - \|L\|)}.$$

Thus, in both algorithms T is bounded by $O(\sqrt{N})$, since in the Monte Carlo algorithm without any absorbing states

$$T < \sqrt{N}\frac{\varepsilon|\log\delta|}{0.6745}$$

and in the Monte Carlo algorithm with absorbing states

$$T \leq \sqrt{N}\frac{\varepsilon}{0.6745}.$$

Results in (Dimov, I., 1993) show that $T \approx \sqrt{N}$, for sufficiently large N.

A Refined Iterative Monte Carlo Approach for Linear Systems and Matrix Inversion Problem

In this part of the course a refined approach of the iterative Monte Carlo algorithms for the well known matrix inversion problem will be presented. The algorithms are based on special techniques of iteration parameter choice (refined stop-criteria), which permits control of the convergence of the algorithm for any row (column) of the matrix using a fine iterative parameter. The choice of this parameter is controlled by $a\ posteriori$ criteria for every Monte Carlo iteration. The algorithms under consideration are also well parallelized.

Formulation of the Problem

Here we deal again with Monte Carlo algorithms for calculating the inverse matrix A^{-1} of a square matrix A, i.e.

$$AA^{-1} = A^{-1}A = I,$$

where I is the identity matrix.

Consider the following system of linear equations:

$$Au = b, \tag{76}$$

where

$$A \in \mathbb{R}^{n\times n}; \quad b, u \in \mathbb{R}^{n\times 1}.$$

The inverse matrix problem is equivalent to solving m-times the problem (76), i.e.

$$Ac_j = b_j, \;\; j = 1, \ldots, n \tag{77}$$

where

$$b_j \equiv e_j \equiv (0, \ldots, 0, 1, 0, \ldots, 0)$$

and

$$c_j \equiv (c_{j1}, c_{j2}, \ldots, c_{jn})^T$$

is the j^{th} column of the inverse matrix $C = A^{-1}$.

Here we deal with the matrix $L = \{l_{ij}\}_{ij=1}^{n}$, such that

$$L = I - DA, \tag{78}$$

where D is a diagonal matrix $D = diag(d_1, \dots, d_n)$ and

$$d_i = \frac{\gamma}{a_{ii}}, \;\; \gamma \in (0, 1] \;\; i = 1, \dots, n.$$

The system (76) can be presented in the following form:

$$u = Lu + f, \tag{79}$$

where

$$f = Db.$$

Let us suppose that the matrix A has diagonally dominant property. In fact, this condition is too strong and the presented algorithms work for more general matrices, as it will be shown later. Obviously, if A is a diagonally dominant matrix, then the elements of the matrix L must satisfy the following condition:

$$\sum_{j=1}^{n} |l_{ij}| \leq 1 \qquad i = 1, \dots, n. \tag{80}$$

Refined Iterative Monte Carlo Algorithms

Here refined iterative Monte Carlo algorithms are considered. The first algorithm evaluates every component of the solution u of the following linear algebraic system (76).

Algorithm 1. *1.* **Input** *initial data: the matrix A, the vector $\mathbf{b}$, the constants $\varepsilon\,, \gamma$ and N.*

2. Preliminary calculations (preprocessing):

2.1. **Compute** *the matrix L using the parameter $\gamma \in (0,1]$:*

$$\{l_{ij}\}_{i,j=1}^{n} = \begin{cases} 1-\gamma \ \text{when} & i=j \\ -\gamma \frac{a_{ij}}{a_{ii}} \ \text{when} & i \neq j\,. \end{cases}$$

2.2. **Compute** *the vector $lsum$:*

$$lsum(i) = \sum_{j=1}^{n} |l_{ij}| \quad \textit{for} \quad i = 1,2,\ldots,n.$$

```
  2.3. Compute the transition probability matrix $P = \{p_{ij}\}_{i,j=1}^{n}$, where

            $$p_{ij} = \frac{|l_{ij}|}{lsum(i)}, \;\; i = 1,2,\dots,n \;\; j = 1,2,\dots,n\,.$$

3. For $i_0 := 1$ to $n$ do step 4 and step 5.

4. While $(W < \varepsilon)$ do the trajectory

  4.1. Set initial values $X := 0$ , $W := 1$ ;
  4.2. Calculate $X := X + W f_{i_0}$ ;
  4.3. Generate an uniformly distributed random number $r \in (0,1)$;
  4.4. Set $j := 1$;
  4.5. If $(r < \sum_{k=1}^{j} p_{i_0 k})$ then
    4.5.1. Calculate $W := W sign(l_{i_0 j}) \times lsum(i_0)$ ;
    4.5.2. Calculate $X := X + W f_j$ (one move in trajectory);
    4.5.3. Update the index $i_0 := j$ and go to step 4.3.
               else
    4.5.4. Update $j := j + 1$ and go to step 4.5.
```

5. **Calculate** *the mean value based on N independent trajectories:*

   5.1. **Do** *"N"-times step 4;*
   5.2. **Calculate** $\overline{X}_N$ *and* $u_{i_0} := \overline{X}_N$.

6. **End** *of Algorithm 1.*

Algorithm 1 describes the evaluation of every component of the solution of the problem (76), which is, in fact, linear algebraic system. Algorithm 1 is considered separately, since it (or some of its steps) will be used in algorithms. For finding the corresponding "i"th component of the solution the following functional is used

$$J(u) = (v, u),$$

where $v = e_i = (0, , \dots, 0 \underbrace{1}_{i}, 0, \dots, 0)$.

We consider the general description of the algorithm - the iteration parameter γ is inside the interval $(0, 1]$.

The second algorithm computes the approximation $\hat{C}$ to the inverse matrix $C = A^{-1}$. The algorithm is based on special techniques of iteration parameter choice. The choice of the iteration parameter γ can be controlled by a $posteriori$ criteria for every column of the approximate inverse matrix $\hat{C}$. The every column of this matrix is computed independently using Algorithm 1.

Algorithm 2. *1.* **Input** *initial data: the matrix A, the constant ε, N and the vector $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_l) \in (0,1]^l$.*

2. **For** $j_0 := 1$ **to** n **do**

Calculate *the elements of j_0^{th} column of the approximate matrix $\hat{C}$:*

2.1. **While** $(k \leq l)$ **do**
2.2. **Apply** *Algorithm 1 for $\gamma = \gamma_k$, N and the right-hand side vector $b_{j_0} = (0, \ldots, 0, \underbrace{1}_{j_0}, 0, \ldots, 0)$ to obtain the column - vector $\hat{c}^{\alpha}_{j_0} = (c^k_{1j_0}, \ldots, c^k_{nj_0})$.*

2.3. **Compute** *the l_2-norm of the column - vector $\hat{c}^k_{j0}$:*

$$r^k_{j0} = \sum_{j=1}^{n} \left\{ \sum_{i=1}^{n} a_{ji} c^k_{ij_0} - \delta_{jj_0} \right\}^2 .$$

2.4. **If** $(r^\alpha_{j_0} < r)$ **then**

$\hat{c}_{j_0} := c^k_{j0};$

$r := r^k_{j0}.$

3. **End** *of Algorithm 2.*

Algorithm 2 is based on Algorithm 1 finding different columns of the matrix $\hat{C}$ by using corresponding values of the iteration parameter $\gamma = \gamma_i, i = 1, 2, \ldots, l$. The values of γ_i are chosen such that to minimize the l_2-norm of the following vectors:

$$E_j = A\hat{C}_j - I^T_j, \quad j = 1, 2, \ldots, n, \tag{81}$$

where $I_j = (0, \ldots, 0, \underbrace{1}_{j}, 0, \ldots, 0)$.

The use of the criteria of minimization of the l_2-norm of the vector E_j permits to find better approximation of the error matrix

$$E = A\hat{C} - I.$$

This procedure allows to minimize the norm (for example, the Frobenius norm) of E. In practice, the parameter γ runs a finite numbers of values in the interval $(0, 1]$.

The evaluation of different columns can be realized in parallel and independently.

The algorithm presented above uses a deterministic approach, which is independent of the statistical nature of the algorithm.

Algorithm 3. *1.* **Input** *initial data: the matrix A, the constant N and the vectors $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_l) \in (0, 1]^l$, $\varepsilon = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)$ and $r = (r_1, r_2, \ldots, r_N)$.*

```
2. **While** $(k \le l)$ **do**
    2.1. Step 2 of Algorithm 1 for $\gamma := \gamma_k$.
    2.2. **While** $(i_0 \le n)$ and $(j_0 \le n)$ **do** step 4 and step 5 of Algorithm 1 to
      compute the elements $\hat{c}^{\alpha}_{i_0 j_0}$ for $\gamma = \gamma_k$ , $\varepsilon := \varepsilon_{i_0}$ and the right-hand side
      vector $b_{j_0} := (0, \dots, 0, \underbrace{1}_{j_0}, 0, \dots, 0)$ .
    2.3. **For** $i_0 := 1$ **to** $n$ **do**
        2.3.1. **Calculate**
            $r^{\alpha}_{i_0} = \max_{i \in \{1,2,\dots,n\}} \left| \sum_{j=1}^{n} c_{i_0 j} a_{ji} - \delta_{i_0 i} \right| .$
        2.3.2. **If** $(r^{\alpha}_{i_0} < r_{i_0})$ **then**
            $\hat{c}_{i_0} := c^{k}_{i_0}$;
            $r_{i_0} := r^{k}_{i_0}$.
3. **End** of Algorithm 3.
```

The difference between the last two algorithms is that Algorithm 3 can not be applied in traditional (non-stochastic) iterative algorithms. The traditional algorithms allow one to evaluate the columns of the inverse matrix in parallel, but they do not allow one to obtain their elements independently from each other. The advantage of the Monte Carlo algorithms consists in possibilities to evaluate every element of the inverse matrix in an independent way. This property allows one to apply different iteration approaches for finding the matrix $\hat{C}$ using *a priori* information for the rows of the given matrix A (for example, the ratio of the sum of the modulus of the non-diagonal entrances to the value of the diagonal element).

One has to mention that the computational complexity of Algorithm 3 also depends on "how ill-conditioned" is the given row of the matrix A. The given row A_i of the matrix A is "ill-conditioned", when the condition

$$|a_{ii}| < \sum_{j=1}^{i-1} |a_{ij}| + \sum_{j=i+1}^{n} |a_{ij}|$$

of *diagonally dominating* is not fulfilled (but all the eigenvalues lay inside the unit circle).

Algorithm 3 presented above is very convenient for such matrices since it chooses the value of the iterative parameter γ for every row of the matrix A. As a measure of the ill-conditioning of a given row we use the following parameter:

$$b_i = \sum_{j=1}^{i-1} |a_{ij}| + \sum_{j=i+1}^{n} |a_{ij}| - |a_{ii}|.$$

The possibility to treat non-diagonally dominant matrices increases the set of the problems treated using Monte Carlo algorithms. For finding different rows of the approximation of the inverse matrix $\hat{C}$ different number of moves (iterations) can be used. The number of moves are controlled by a parameter ε. For *a posteriori* criteria we use the minimization of the C-norm of the following row-vector

$$E_i = \hat{C}_i A - I_i, \quad i = 1, \ldots, n,$$

where $\hat{C}_i = (0, \ldots, 0, \underbrace{1}_{i}, 0, \ldots, 0)$.

The use of the criteria of minimization of the C-norm of the vector E_i permits finding better approximation of the error matrix

$$E = \hat{C}A - I. \tag{82}$$

The above mentioned procedure allows the minimization of the norm (for example, the Frobenius norm) of the matrix E.

One can also control the number of moves in the Markov chain (that is the number of iterations) to have a good balance between the stochastic and systematic error (i.e., the truncation error). The problem of balancing of the systematic and stochastic errors is very important when Monte Carlo algorithms are used. It is clear that in order to obtain good results the stochastic error (the probable error) r_N must be approximately equal to the systematic one r_s, that is

$$r_N = O(r_s).$$

The problem of balancing the errors 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. The balancing allows an increase in the accuracy of the algorithm for a fixed computational complexity, because in this case one can control the parameter $E(Y)$ by choosing different lengths of the realizations of the Markov chain. In practice, we choose the absorbing state of the random trajectory using the well known criteria

$$|W| < \varepsilon.$$

Such a criteria is widely used in iterative algorithms, but obviously it is not the best way to define the absorbing states of the random trajectory. It is so, because for different rows of the matrix A the convergence of the corresponding iterative process (and, thus, the truncation error) may be different. If the rate of convergence is higher it is possible to use a higher value for the parameter ε and to cut the random trajectory earlier than in the case of lower convergence. Our approach permits use of different stop-criteria for different random trajectories, which allows the optimization of the algorithm in the sense of balancing errors.

Discussion of the numerical results

As an example we consider matrices arising after applying the mixed finite element algorithm for the following boundary value problem

$$\left| \begin{array}{l} -\text{div}(a(x)\underline{\nabla}p) = f(x), \text{ in } \Omega \\ p = 0, \text{ on } \partial\Omega, \end{array} \right. \tag{83}$$

where $\underline{\nabla} w$ denotes the gradient of a scalar function w, $div\ \underline{v}$ denotes the divergence of the vector function $\underline{v}$ and $a(x)$ is a diagonal matrix whose elements satisfy the requirements $a_i(x) \geq a_0 > 0,\ i = 1, 2$.

We set

$$\underline{u} \equiv (u_1, u_2) = a(x)\underline{\nabla}p, \quad \alpha_i(x) = a_i(x)^{-1}, i = 1, 2.$$

Let us consider the spaces $\underline{V}$ and W defined by

$$\begin{array}{rcl} \underline{V} & = & \underline{H}(div;\Omega) = \{\underline{v} \in L^2(\Omega)^2 \ :\ div\ \underline{v} \in L^2(\Omega)\}, \\ W & = & L^2(\Omega) \end{array}$$

provided with the norms

$$\begin{aligned} \|\underline{v}\|_{\underline{V}} &\equiv \|\underline{v}\|_{\underline{H}(div;\Omega)} = (\,\|\underline{v}\|_{0,\Omega}^2 + \|div\ \underline{v}\|_{0,\Omega}^2)^{1/2} \ \text{ and} \\ \|w\|_W &= \|w\|_{L^2(\Omega)} = \|w\|_{0,\Omega} \end{aligned}$$

respectively.

Then the mixed variational formulation of the problem (83) is given by characterizing the pair $(\underline{u}, p)$, as the solution of

$$\left| \begin{array}{ll} a(\underline{u},\underline{v}) + b(\underline{v},p) = 0, & \forall \underline{v} \in \underline{V}; \\ b(\underline{u},w) = -(f,w), & \forall w \in W, \end{array} \right. \tag{84}$$

where

$$a(\underline{u},\underline{v}) = (\alpha u_1, v_1) + (\alpha u_2, v_2), \quad b(\underline{u},w) = (div\underline{u}, w)$$

and $(\cdot,\cdot)$ indicated the inner product in $L^2(\Omega)$.

The mixed finite element approximation of the problem (84) in the Raviart-

Thomas spaces leads to the following linear algebraic system:

$$Ku = \begin{pmatrix} A_1 & 0 & B_1 \\ 0 & A_2 & B_2 \\ B_1^T & B_2^T & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ p \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -f \end{pmatrix}, \tag{85}$$

where A_i are $n \times n$ matrices, B_i are $n \times n_1$ matrices $(n_1 < n)$, $u_i \in \mathbb{R}^n$ and $p, f \in \mathbb{R}^{n_1}$, $i = 1, 2$.

If $A_i^{-1}\,(i = 1, 2)$ is obtained then the system (85) becomes

$$Bp = f,$$

where

$$B = B_1^T A_1^{-1} B_1 + B_2^T A_2^{-1} B_2$$

Thus we reduce the $\{2m + m_1\}$-dimensional linear algebraic system to the m_1-dimensional system.

Table 6: Connection between ε and the parameter γ. Here $m = 16$, $n = 24$.

column	l_1 norm					C norm			
number	$\varepsilon =$	0.05	0.01	0.005	0.001	0.05	0.01	0.005	0.001
1		1	0.9	0.6	0.2	0.5	0.1	1	1
2		0.4	0.8	0.9	0.8	0.5	0.9	0.6	1
3		1	0.8	0.8	0.9	0.5	1	0.3	0.1
4		0.5	1	0.7	0.7	0.3	0.3	1	0.9
5		0.9	0.5	0.9	0.9	1	1	0.9	0.8
6		0.8	0.1	0.6	0.6	1	0.8	0.9	0.8
7		0.5	0.1	0.9	0.9	0.8	0.4	0.9	1
8		0.5	0.1	0.6	0.9	0.8	0.8	0.3	1
9		0.5	0.1	0.6	0.6	0.6	1	1	0.2
10		0.5	0.1	0.6	0.3	0.6	1	0.4	0.5
11		0.5	0.1	0.6	0.3	0.5	0.1	1	0.5
12		0.5	0.1	0.6	0.3	0.7	0.8	1	0.8
13		0.5	0.1	0.6	0.3	1	1	0.4	0.9
14		0.5	1	0.6	0.3	0.9	0.9	0.4	1
15		1	0.1	0.8	0.9	1	1	1	0.4
16		0.9	0.3	0.6	0.1	1	1	0.9	0.6

As a basic test example for applying Algorithm 3 a matrix of of size 7 is used. The size of the matrix is relatively small, because our aim was only to demonstrate how Algorithm 3 works. Here we also have to mention that the computational complexity of the algorithm practically does not depend of the size of the matrix. Using the technique from (Dimov, Karaivanova 1996) it is possible to show that the computational complexity of our algorithms depends linearly of the mean value of the number of non-zero entrances per row. This is very important, because it means that very large sparse matrices could be treated efficiently using the algorithms under consideration.

During the numerical tests we control the Frobenius norm of the matrices, defined by

$$\| A \|_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}^2.$$

Some of the numerical results performed are shown in Figures 5 to 8, and also provided by Table 6. In all figures the value of the Frobenius norm is denoted by F.N., the number of realizations of the random variable (i.e., the number of random trajectories) is denoted by N and the value of the stop-criteria is denoted by ε. In **Algorithm 3** we use n values of ε (in our case $n = 7$).

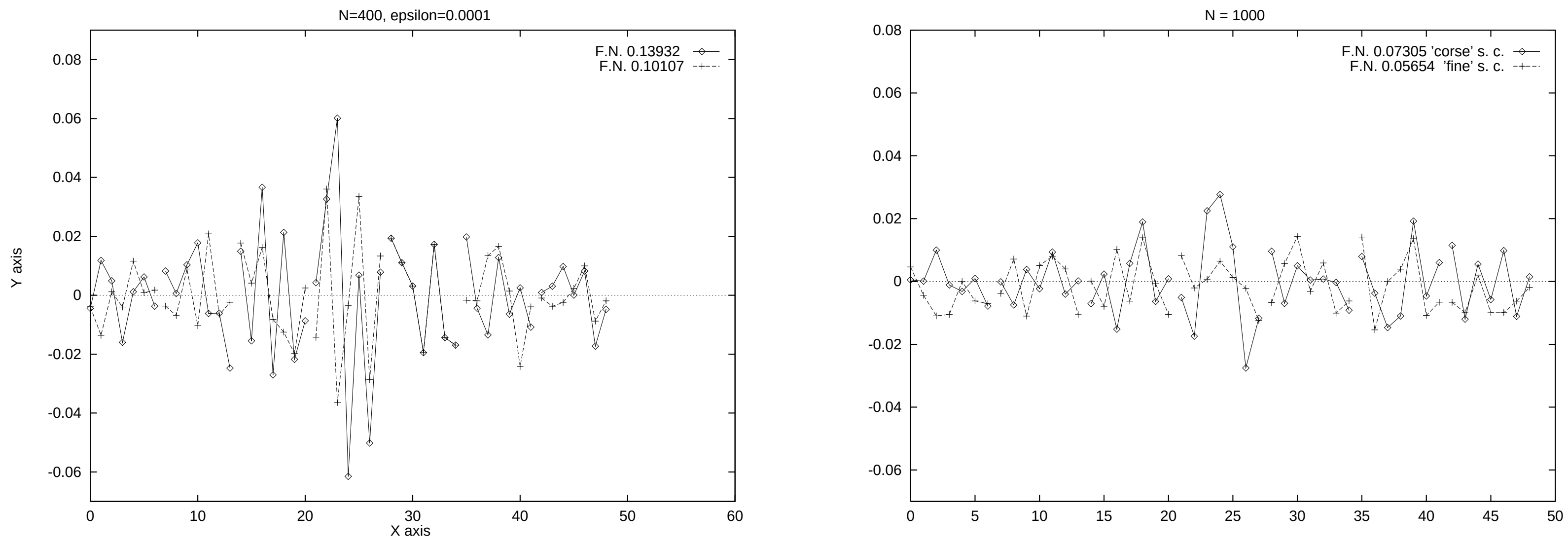
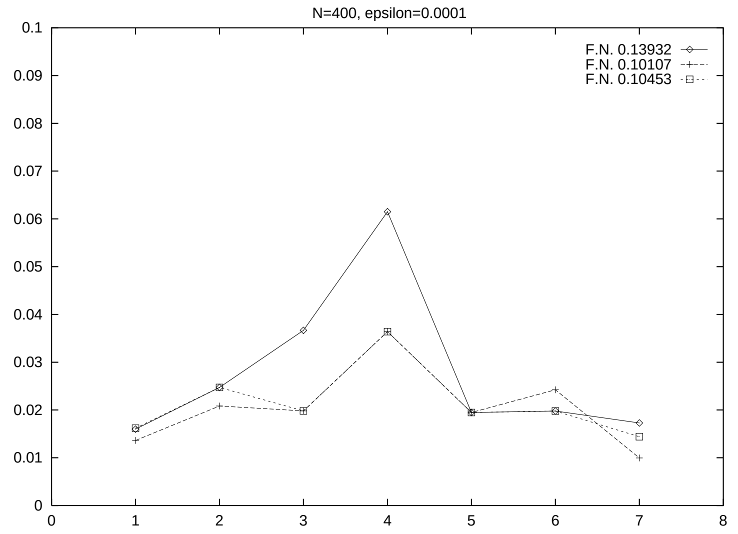


Figure 5: Frobenius norm: (a) non-balanced case; (b) balanced case.

Figure 5 presents the values of the error matrix (82) in the both cases under consideration – coarse stop criteria (a) and fine stop criteria (b). The first set of connected points corresponds to values of the first row of the error matrix, the second set – to the second row of the same matrix, etc.



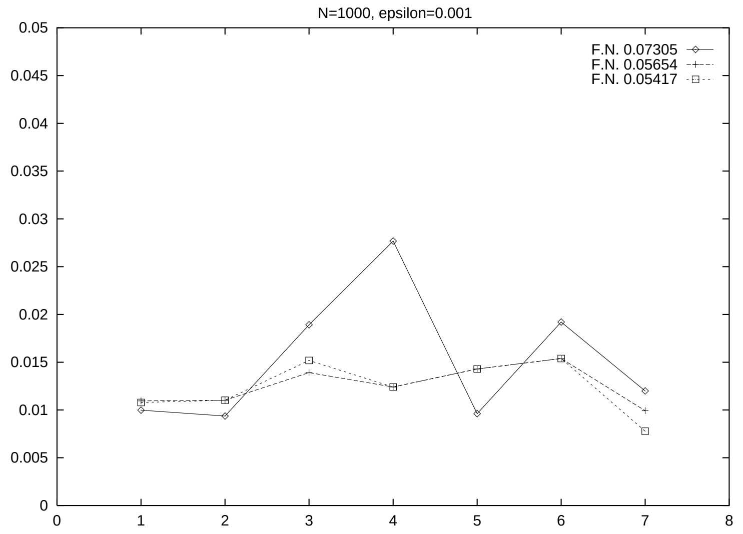


Figure 6: Balancing: (a) Non-controlled balance; (b) Controlled balance.

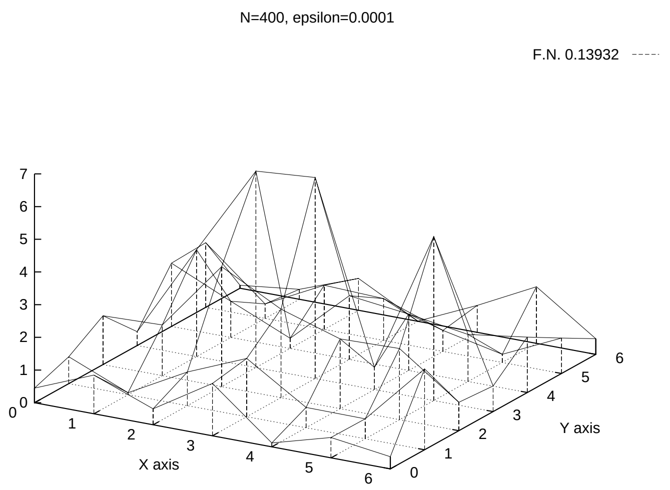
When the coarse stop criteria is used (Figure 5 (a)) $\varepsilon = 0.0001$. When the fine stop criteria is used (Figure 5 (b)) different values of ε are applied such that the computational complexity is smaller (in comparison with the case if the coarse stop criteria) (see, also Table 6). The values of the Frobenius norm for both cases when the number of realizations N is equal to 400 are also given. For such number of realizations the stochastic error is relatively large in comparison with the systematic one. So, the results on Figure 5(a) correspond to the non-balanced case. The similar results, but for the case of $N = 1000$

and $\varepsilon = 0.001$ (for the coarse stop criteria) are presented on Figure 5(b). One can see, that

- ε is 10 times large then in the previous case, but the Frobenius norm is about two times smaller, because the number of realizations is larger.

The results presented in Figure 5(a) and Figure 5(b) show the statistical convergence of the algorithm, i.e. the error decreases when N increases (even in the case when the parameter ε increases).

These results show how important is to have a good balancing between the stochastic and systematic error. The computational effort for the cases presented in Figure 5(a) and Figure 5(b) is approximately equal, but the results in the case of Figure 5(b), when we have a good balancing are almost 2 times better.



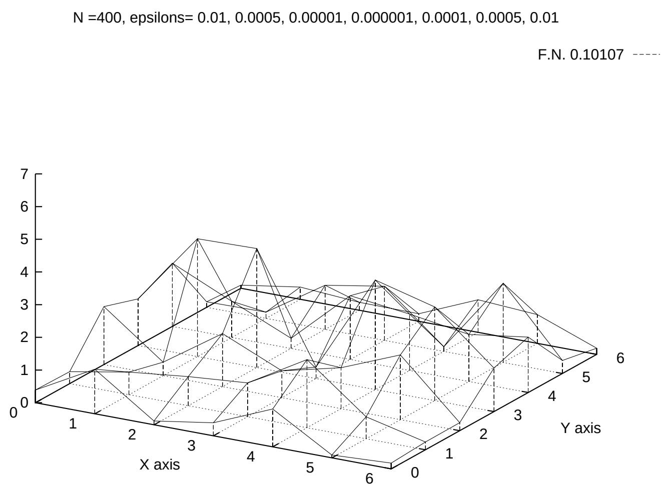
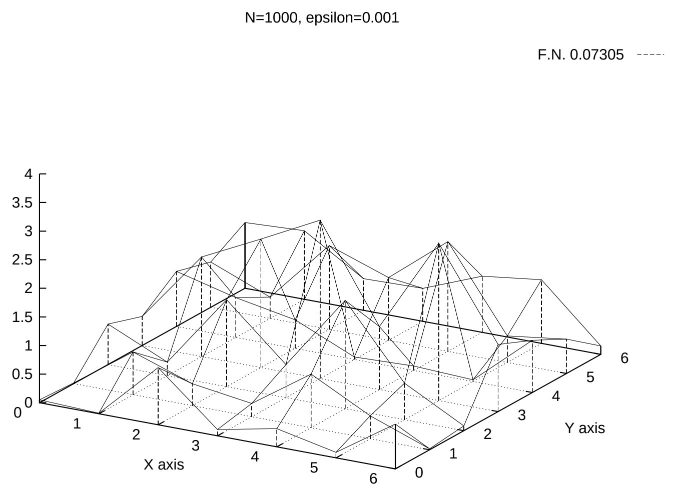


Figure 7: Use of different fine stop-criteria: (a) Coarse stop-criteria; (b) Fine stop-criteria.



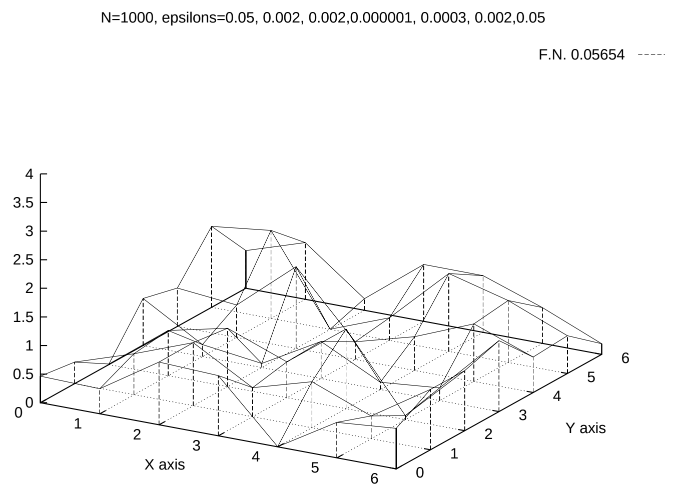


Figure 8: Controlled balancing: (a) Coarse stop-criteria; (b) Fine stop-criteria.

Let us discuss the results presented in Figures 6(a) and 6(b). Here instead of elements of the error matrix the maximum of the modulo element for every row are shown. If the computational complexity for a constant ε is denoted by R_c and and the computational complexity when different values of $\varepsilon = \varepsilon_i,\ i = 1, \ldots, n$ is denoted by R_f we consider the case when

$$R_c \geq R_f = 1.$$

The results presented in Figures 6(a) and 6(b) show that apart from the smaller computational complexity R_f of the fine stop criteria algorithm it also gives better results than the coarse stop criteria algorithm with complexity R_c. This fact is observed in both cases - balanced (Figure 6(a)) and non-balanced (Figure 6(b)):

- the variations of the estimations are smaller when the balancing is better;
- the Frobenius norm is smaller, when the control “row per row” is realized.

Figures 7(a) and 7(b) present test results for the modulo of every element of the error matrix (82) when the coarse stop criteria and fine stop criteria respectively are used in the non-balanced case.

One can see, that:

- the Frobenius norm of the estimate in the case of fine stop criteria is about 1.4 times smaller than the corresponding value for the coarse stop criteria, and;

- the variances of the estimate of the case of fine stop criteria are smaller.

Figures 8(a) and 8(b) show the corresponding results as in Figures 7(a) and 7(b) in the balanced case. One can make the same conclusion as in the non balanced case, but here

- the Frobenius norm is almost 2 times smaller.

Table 7: Computational complexity

	non balanced case		balanced case	
γ	*coarse s. c.*	*fine s. c.*	*coarse s. c.*	*fine s. c.*
0.2	1.0806	1	1.0368	1
0.4	1.0903	1	1.0351	1
0.6	1.0832	1	1.0348	1
0.8	1.0910	1	1.0360	1
1	1.0862	1	1.0342	1
generally	1.0848	1	1.0358	1

Results presented in Table 7 show how the computational complexity depends on the parameter γ for the balanced and non balanced cases. One can see that the application of fine stoping criteria makes the balanced algorithm about 3.5% more efficient than the algorithm in which the course stoping criteria is used. At the same time for the non balanced case the fine stoping criteria algorithm is approximately 8.5% more efficient than the course stoping criteria algorithm.

Conclusion

An iterative Monte Carlo algorithm is presented and studied. This algorithm can be applied for solving of inverse matrix problems.

The following conclusion can be done:

- Every element of the inverse matrix A^{-1} can be evaluated independently from the other elements (this illustrates the inherent parallelism of the algorithms under consideration);
- Parallel computations of every column of the inverse matrix A^{-1} with different iterative procedures can be realized;
- It is possible to optimize the algorithm using error estimate criterion “column by column”, as well as “row by row”;
- The balancing of errors (both systematic and stochastic) allows to increase the accuracy of the solution if the computational effort is fixed or to reduce the computational complexity if the error is fixed.

The studied algorithm is easily programmable and parallelizable and can be efficiency implemented on MIMD (Multi Instruction – Multi data)-machines.

IV. Monte Carlo Methods for Boundary-Value Problems (BVP)

There are essentially two approaches to numerically solving elliptic equations. The first one is the so-called *grid approach*, while the second one might be called the *grid-free approach*. Here we consider both approaches.

Let $\Omega \subset \mathbb{R}^d$ be a bounded domain with a boundary $\partial\Omega$.

The following notations are used:

$x = (x_{(1)}, x_{(2)}, \dots, x_{(d)})$ is a point in $\mathbb{R}^d$;

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

We consider the linear boundary value problem

$$Lu \equiv \sum_{|\alpha|\leq 2m} a_\alpha(x) D^\alpha u(x) = -f(x), \qquad x \in \Omega \tag{86}$$

$$u(x) = \varphi(x), \qquad x \in \partial\Omega\,, \tag{87}$$

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

We use the following definition of *ellipticity*:

Definition 5. *The equation*

$$\sum_{|\alpha|\leq 2m} a_\alpha(x) D^\alpha u(x) = -f(x)$$

is called elliptic in a domain Ω if

$$\sum_{|\alpha|\leq 2m} a_\alpha(x) \xi_{\alpha_1}\xi_{\alpha_2}\ldots\xi_{\alpha_d} \neq 0 \;\text{ when } |\xi| \neq 0$$

holds for every point $x \in \Omega$. The corresponding operator $\sum_{|\alpha| \leq 2m} a_\alpha(x) D^\alpha$ is called elliptic in Ω.

Assume that $f(x)$, $\varphi(x)$, and the boundary $\partial\Omega$ satisfy conditions ensuring that the solution of the problem (86, 87) exists and is unique (Jost, J., 2002; Miranda, C., 1955).

We shall study Monte Carlo algorithms for calculating linear functionals of the solution of the problem (86, 87)

$$J(u) = (h, u) = \int_\Omega u(x)h(x)dx, \tag{88}$$

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

For many applications $\mathbf{X} = \mathbf{L}_1$ and thus $\mathbf{X}^* = \mathbf{L}^\infty$, or $\mathbf{X} = \mathbf{L}_2$, $\mathbf{X}^* = \mathbf{L}_2$.

There are two approaches for calculating (88). The first approach uses a discretization of the problem (86, 87) on a mesh and solves the resulting linear algebraic system, which approximates the original problem (86, 87). This approach leads to the so-called *grid Monte Carlo algorithm*, or *grid walk algorithm*. The second approach - the *grid-free approach* - uses an integral representation for the problem (86, 87).

Grid Monte Carlo Algorithm

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

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

$$(L_h u_h)_i = \sum_{x_j \in P_k(x_i)} a_h(x_i, x_j) u_h(x_j) \ , \tag{89}$$

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

Since L is a linear differential operator, after the discretization of (89), the following system of linear equations arises

$$Au = b, \tag{90}$$

where $b = (b_1, \ldots, b_n)^T \in \mathbb{R}^{n\times 1}$ is an n-dimensional vector and $A \in \mathbb{R}^{n\times n}$ is an $n \times n$-dimensional matrix.

Grid-Free Monte Carlo Algorithms

Consider two approaches for constructing grid-free Monte Carlo algorithms. The first one consists in obtaining a *global integral representation* both on the boundary and on the domain.

Let us consider an example of the following linear elliptic BVP:

$$\Delta u(x) - c^2 u(x) = -\varphi(x), \; x \in \Omega \tag{91}$$

$$u(x) = \psi(x), \; x \in \partial\Omega, \tag{92}$$

where Δ is the Laplacian and the functions $\varphi(x)$, $\psi(x)$ and the boundary satisfy all conditions, which provide the existence of a unique solution of the problem (91, 92).

From the theory of fundamental solutions it follows that the solution of the problem (91, 92) can be represented as the integral equation (44) (see, Bitzadze, A.V., 1982; Ermakov, S.M., 1982), where

$$k(x,y) = \begin{cases} \dfrac{cd(x)}{\sinh[cd(x)]}\delta(y-x) & \text{, when } \ x \ \in \Omega \setminus \partial\Omega \\ 0 & \text{, when } \ x \in \partial\Omega \end{cases}$$

$$f(x) = \begin{cases} \dfrac{1}{4\pi}\displaystyle\int \dfrac{\sinh((d(x) - |y-x|)c}{|y-x| \sinh[cd(x)]}\varphi(y)dy & , \ \text{when} \ \ x \ \in \Omega \setminus \partial\Omega \\ \psi(x) & , \ \text{when} \ \ x \in \partial\Omega \end{cases}$$

and $d = d(x)$ is the distance from x to the boundary $\partial\Omega$.

It will it be necessary to calculate the functional (88), where u is the solution of the problem (91, 92) and h is a given function.

This representation permits the use of a random variable for calculating the functional (88). Unfortunately, this approach is not successful when one deals with more complicated operators for which it is impossible to find an integral representation.

The second grid-free Monte Carlo approach 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 Ω (for example - ball, sphere, ellipsoid) is used.

Consider the elliptic boundary value problem:

$$Mu = -\phi(x), \ x \in \Omega, \ \Omega \subset \mathbb{R}^3 \tag{93}$$

$$u = \psi(x), \ x \in \partial\Omega, \tag{94}$$

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 $\mathcal{A}^{(k,\lambda)}$:

Definition 6. *The domain Ω belongs to the class $\mathcal{A}^{(k,\lambda)}$ if for any point $x \in \partial\Omega$ (from the boundary $\partial\Omega$) the boundary $\partial\Omega$ can be presented as a function $z_3 = \sigma(z_1, z_2)$ in the neighborhood of x for which $\sigma^{(k)}(z_1, z_2) \in \mathbf{H}^{\lambda}(\alpha; \partial\Omega)$, i.e.*

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

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

If in the closed domain $\bar{\Omega} \in \mathcal{A}^{(1,\lambda)}$ the coefficients of the operator M satisfy the conditions $b_j,\ c(x) \in \mathbf{H}^{\lambda}(\alpha; \bar{\Omega}),\ c(x) \leq 0$ and $\phi \in \mathbf{H}^{\lambda}(\alpha; \Omega) \cap \mathbf{C}(\bar{\Omega})$, $\psi \in \mathbf{C}(\partial\Omega)$, the problem (93, 94) has a unique solution $u(x)$ in $\mathbf{C}^2(\Omega) \cap \mathbf{C}(\bar{\Omega})$. The conditions for uniqueness of a solution can be found in (Jost, J., 2002, p. 179).

We obtain an integral representation of the solution $u(x)$. This representation allows for the use of the random variable for calculating the functional (88).

Local Integral Representation

We use the grid-free Monte Carlo approach to estimate the functional (88). This approach is based on the use of a local integral representation of the solution $u(x)$ in the problem (93, 94). The representation uses the Green's function approach for standard domains, lying inside the domain Ω. The initial step in studying the grid-free Monte Carlo approach is to obtain an integral representation of the solution in the form:

$$u(x) = \int_{B(x)} k(x,y)u(y)dy + f(x) \tag{95}$$

assuming that such a representation exists.

The iterative Monte Carlo process converges when the condition

$$\| K(u) \|_{L_1} = \max_{x\in\Omega} \int_{\Omega} | k(x,y) | \, dy \leq q < 1 \tag{96}$$

holds.

For the existence of the integral representation, (95) might be obtained using the result of C. Miranda, 1955, taking into consideration that the domain $B(x)$ belongs to the space $\mathcal{A}^{(1,\lambda)}$ 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 following Lemma holds:

Lemma 1. *Let the components of the vector-function* $\mathbf{b}(x)$ *satisfy the conditions* $b_j(x) \in C^{(1)}(\Omega),\ (j = 1, 2, 3)$ *and* $div\mathbf{b}(x) = 0$.

Then the adjoint operator M^* *applied on functions* $v(x)$, *where* $v \in C^2(\Omega)$ *and*

$$\frac{\partial v(x)}{\partial x_{(i)}} = v(x) = 0 \ \textit{for any} \ \ x \in \partial\Omega,\ i = 1, 2, 3$$

has the following form:

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

Proof 10. *Let us show that M^* is an adjoint operator to M, i.e. we have to prove that*

$$\int_\Omega v(x) M u(x) dx = \int_\Omega u(x) M^* v(x) dx. \tag{97}$$

To prove (97) we use Green's formulas:

$$\begin{aligned}\int_\Omega u(x) \sum_{i=1}^{3} \frac{\partial v(x)}{\partial x_{(i)}} dx &= -\int_\Omega v(x) \sum_{i=1}^{3} \frac{\partial u(x)}{\partial x_{(i)}} dx \\ &+ \int_{\partial\Omega} \sum_{i=1}^{3} u(x) v(x) n_i d_x S \end{aligned} \tag{98}$$

and

$$\int_\Omega u(x)\Delta v(x)dx = -\int_\Omega \textit{grad}\, u(x)\textit{grad}\, v(x)dx + \int_{\partial\Omega} u(x)\sum_{i=1}^{3} n_i \frac{\partial v(x)}{\partial x_{(i)}} d_x S,$$

where

$$\Delta = \sum_{i=1}^{3} \frac{\partial^2}{\partial x_{(i)}^2}, \quad \textit{div}\,\mathbf{b}(x) = \sum_{i=1}^{3} \frac{\partial b_i(x)}{\partial x_{(i)}},$$

$$\textit{grad}\, u(x) \equiv \left(\frac{\partial u(x)}{\partial x_{(1)}}, \frac{\partial u(x)}{\partial x_{(2)}}, \frac{\partial u(x)}{\partial x_{(3)}} \right),$$

and $\mathbf{n} \equiv (n_1, n_2, n_3)$ *is the exterior normal for the boundary* $\partial\Omega$.

Taking into consideration that

$$\textit{div}\,\mathbf{b}(x) = 0 \ \text{ and } \ \frac{\partial v(x)}{\partial x_{(i)}} = v(x) = 0 \ \textit{ for any } \ \ x \in \partial\Omega, \ i = 1, 2, 3,$$

we have

$$\int_\Omega v(x)Mu(x)dx = \int_\Omega v(x)\left(\Delta u(x) + \mathbf{b}(x)\mathit{grad}\, u(x) + c(x)u(x)\right)dx$$

$$= -\int_\Omega \mathit{grad}\, v(x)\mathit{grad}\, u(x)dx + \int_{\partial\Omega} v(x)\sum_{i=1}^{3} n_i \frac{\partial u(x)}{\partial x_{(i)}} d_x S$$

$$+ \int_\Omega v(x)\mathbf{b}(x)\mathit{grad}\, u(x)dx + \int_\Omega v(x)c(x)u(x)dx$$

$$= -\int_\Omega \mathit{grad}\, v(x)\mathit{grad}\, u(x)dx + \int_\Omega v(x)\sum_{i=1}^{3} b_i(x)\frac{\partial u(x)}{\partial x_{(i)}} dx + \int_\Omega v(x)c(x)u(x)dx.$$

On the other hand

$$\int_\Omega u(x)M^*v(x)dx = \int_\Omega u(x)\left[\Delta v(x) - \mathbf{b}(x)\mathit{grad}\, v(x) + c(x)v(x)\right] dx$$

$$= -\int_\Omega \mathit{grad}\, u(x)\mathit{grad}\, v(x)dx + \int_{\partial\Omega} u(x)\sum_{i=1}^{3} n_i \frac{\partial v(x)}{\partial x_{(i)}} d_x S$$

$$- \int_\Omega u(x)\mathbf{b}(x)\mathit{grad}\, v(x)dx + \int_\Omega u(x)c(x)v(x)dx$$

$$= -\int_\Omega \mathit{grad}\, u(x)\mathit{grad}\, v(x)dx - \int_\Omega u(x)\sum_{i=1}^{3} b_i(x)\frac{\partial v(x)}{\partial x_{(i)}} dx$$

$$+ \int_\Omega u(x)c(x)v(x)dx$$

$$= -\int_\Omega \mathit{grad}\, u(x)\mathit{grad}\, v(x)dx + \int_\Omega v(x)\sum_{i=1}^{3} \frac{\partial(u(x)b_i(x))}{\partial x_{(i)}} dx$$

$$- \int_{\partial\Omega} \sum_{i=1}^{3} n_i b_i(x)u(x)v(x)dx + \int_\Omega v(x)c(x)u(x)dx$$

From the last result there follows the proof of the lemma.

The Levy's function for the problem (93, 94) is

$$L_p(y,x) = \mu_p(R)\int_r^R (1/r - 1/\rho)p(\rho)d\rho, \;\; r \le R, \tag{99}$$

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 the 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 (99) defines a family of functions.

We seek a choice of $p(\rho)$ which leads to a representation of type (95). Moreover, the kernel of the integral transform should be a transition density function, i.e. $k(x, y) \geq 0$.

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

Denote by $B(x)$ the ball:

$$B(x) = B_R(x) = \{y : r =\mid y - x \mid\leq R(x)\}, \tag{100}$$

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

For the Levy's function $L_p(y, x)$ the following representation holds (see, Miranda, C., 1955):

$$\begin{aligned} u(x) &= \int_{B(x)} \big(u(y)M_y^* L_p(y,x) + L_p(y,x)\phi(y)\big)\, dy \\ &+ \int_{\partial B(x)} \sum_{i=1}^{3} n_i \left[\left(\frac{L_p(y,x)\partial u(y)}{\partial y_{(i)}} - \frac{u(y)\partial L_p(y,x)}{\partial y_{(i)}}\right)\right. \\ &- \left. b_i(y)u(y)L_p(y,x)\right] d_y S, \end{aligned} \tag{101}$$

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

Formula (101) holds for any domain $T(x) \in \mathcal{A}^{(1,\lambda)}$ contained in Ω.

Obviously, $B(x) \in \mathcal{A}^{(1,\lambda)}$ and therefore for every ball lying inside the domain Ω the representation (101) holds.

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

$$\begin{aligned} M_y^* G(x,y) &= -\delta(x-y),\ y \in \Omega \setminus \partial\Omega \setminus \{x\}, \\ G(x,y) &= 0,\ \ y \in \partial\Omega,\ x \in \Omega \setminus \partial\Omega. \end{aligned}$$

The Green's function is the Levy's function, $L_p(y,x)$, for which (93, 94) hold.

Under the condition $L_p(y,x) = G(x,y)$ from (101) it is possible to get the integral representation:

$$u(x) = \int_{B(x)} G(x,y) f(y) dy - \int_{\partial B(x)} \sum_{i=1}^{3} n_i \frac{\partial G(x,y)}{\partial y_{(i)}} u(y) d_y S. \qquad (102)$$

Representation (102) is the basis for the Monte Carlo method.

For achieving this aim it is necessary to have a non-negative integral kernel. Next we show 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 2. *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

$$k \geq \max_{x \in \Omega} \mid \mathbf{b}(x) \mid + R \max_{x \in \Omega} \mid c(x) \mid \tag{103}$$

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

Proof 11. *The condition*

$$L_p(y,x) = 0, \quad \textit{for any} \quad y \in \partial B(x)$$

obviously holds. It follows from (99), (100), since if $y \in \partial B(x)$, then $r = R$ and $L_p(y,x) = 0$.

The condition

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

can be checked immediately. Indeed,

$$\begin{aligned}
\frac{\partial L_p(y,x)}{\partial y_{(i)}} &= \frac{\partial L_p}{\partial r}\frac{\partial r}{\partial y_{(i)}} = \mu_p(R)\frac{\partial}{\partial r}\left(\int_r^R (1/r - 1/\rho)p(\rho)d\rho\right)\frac{\partial r}{\partial y_{(i)}} \\
&= \mu_p(R)\frac{\partial}{\partial r}\left(\frac{1}{r}\int_r^R p(\rho)d\rho - \int_r^R \frac{1}{\rho}p(\rho)d\rho\right)\frac{\partial r}{\partial y_{(i)}} \\
&= \mu_p(R)\left[-\frac{1}{r^2}\int_r^R p(\rho)d\rho + \frac{1}{r}(-p(r)) - \left(-\frac{1}{r}p(r)\right)\right]\frac{\partial r}{\partial y_{(i)}} \\
&= \mu_p(R)\left(-\frac{1}{r^2}\int_r^R p(\rho)d\rho\right)\frac{\partial r}{\partial y_{(i)}}.
\end{aligned}$$

Taking into consideration that $\frac{\partial r}{\partial y_{(i)}} = \frac{-(x_{(i)}-y_{(i)})}{r}$ *one can get*

$$\frac{\partial L_p(y,x)}{\partial y_{(i)}} = \mu_p(R)\frac{(x_{(i)}-y_{(i)})}{r^3}\int_r^R p(\rho)d\rho. \tag{104}$$

The last expression vanishes when $r = R$*, i.e. for every boundary point* $y \in \partial B(x)$*. Thus we obtain*

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

Now calculate $M_y^* L_p(y,x)$*. The operator* M_y^* *has the following form:*

$$M_y^* = \sum_{i=1}^{3}\left(\frac{\partial^2}{\partial y_{(i)}^2}\right) - \sum_{i=1}^{3}\left(b_i(y)\frac{\partial}{\partial y_{(i)}}\right) + c(y)$$

and $M_y^* L_p(y,x)$ has the form:

$$\begin{aligned} M_y^* L_p(y,x) &= \sum_{i=1}^{3} \left(\frac{\partial^2 L_p(y,x)}{\partial y_{(i)}^2} \right) \\ &- \sum_{i=1}^{3} \left(b_i(y) \frac{\partial L_p(y,x)}{\partial y_{(i)}} \right) + c(y) L_p(y,x). \end{aligned} \tag{105}$$

The second term of (105) is calculated using (104), i.e.

$$\sum_{i=1}^{3} b_i(y) \frac{\partial L_p(y,x)}{\partial y_{(i)}} = \mu_p(R) \sum_{i=1}^{3} b_i(y) \frac{(x_{(i)} - y_{(i)})}{r^3} \int_r^R p(\rho) d\rho. \tag{106}$$

Calculate the first term in (105). That can be done easily when we use spherical coordinates:

$$y_{(1)} - x_{(1)} = r \, \sin\theta \cos\varphi, \quad y_{(2)} - x_{(2)} = r \, \sin\theta \sin\varphi, \quad y_{(3)} - x_{(3)} = r \, \cos\theta,$$

where $0 < r < R(x), \quad \theta \in [0,\ \pi)$ *and* $\varphi \in [0,\ 2\pi)$.

Thus the Laplacian

$$\Delta_y = \sum_{i=1}^{3} \left(\frac{\partial^2}{\partial y_{(i)}^2} \right)$$

written in spherical coordinates has the following form (Tikchonov, A.N., p. 282):

$$\Delta_{r,\theta,\varphi} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 \frac{\partial}{\partial r}\right) + \frac{1}{r^2 \sin\theta}\frac{\partial}{\partial \theta}\left(\sin\theta \frac{\partial}{\partial \theta}\right) + \frac{1}{r\sin^2\theta}\frac{\partial^2}{\partial \varphi^2}.$$

The Levy's function in spherical coordinates depends on the radius r, *(see,* *(99)). Thus,*

$$\begin{aligned}
\Delta_y L_p(y,x) &= \Delta_{r,\theta,\varphi} L_p(r) = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 \frac{\partial L_p(r)}{\partial r}\right) \\
&= \mu_p(R)\frac{1}{r^2}\frac{\partial}{\partial r} r^2 \frac{\partial}{\partial r}\left(\int_r^R (1/r - 1/\rho)p(\rho)d\rho\right) \\
&= \mu_p(R)\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\left(-\frac{1}{r^2}\right)\int_r^R p(\rho)d\rho\right) \\
&= \mu_p(R)\left(-\frac{1}{r^2}\right)\frac{\partial}{\partial r}\int_r^R p(\rho)d\rho = \mu_p(R)\frac{p(r)}{r^2}. \qquad (107)
\end{aligned}$$

Taking into consideration (105), (106) we obtain:

$$\begin{aligned} M_y^* L_p(y,x) &= \mu_p(R)\frac{p(r)}{r^2} - \mu_p(R)c(y)\int_r^R \frac{p(\rho)}{\rho}d\rho \\ &+ \frac{\mu_p(R)}{r^2}\left[c(y)r + \sum_{i=1}^{3} b_i(y)\frac{y_{(i)} - x_{(i)}}{r}\right]\int_r^R p(\rho)d\rho. \end{aligned}$$

Next we prove that $M_y^* L_p(y,x)$ is non-negative for every point of the ball $B(x)$. Write $M_y^* L_p(y,x)$ in the following form:

$$M_y^* L_p(y,x) = \frac{\mu_p(R)}{r^2}\Gamma_p(y,x),$$

where

$$\begin{aligned}\Gamma_p(y,x) &= p(r) + c(y)r\left(\int_r^R p(\rho)d\rho - \int_r^R \frac{p(\rho)r}{\rho}d\rho\right) \\ &+ \sum_{i=1}^{3} b_i(y)\frac{y_{(i)} - x_{(i)}}{r}\int_r^R p(\rho)d\rho.\end{aligned}$$

It is necessary to show that for all $y \in B(x)$ *the function* $\Gamma_p(y,x)$ *is non-negative. From the condition* $c(y) \le 0$ *it follows that*

$$\begin{aligned}\Gamma_p(y,x) &= p(r) - \left|c(y)r\left(\int_r^R p(\rho)d\rho - \int_r^R \frac{p(\rho)r}{\rho}d\rho\right)\right| \\ &+ \sum_{i=1}^{3} b_i(y)\frac{y_{(i)} - x_{(i)}}{r}\int_r^R p(\rho)d\rho \ge 0. \qquad (108)\end{aligned}$$

So, it is necessary to prove (108). For $p(r) = e^{-kr}$ *we have*

$$p(r) \geq e^{-kr} - e^{-kR} = k \int_r^R p(\rho) d\rho.$$

Choosing

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

one can obtain

$$\begin{aligned} p(r) &\geq \left(\max_{x\in\Omega} \mid \mathbf{b}(x) \mid + R\max_{x\in\Omega} \mid c(x) \mid\right) \int_r^R p(\rho)d\rho \\ &\geq \mid c(y) \mid r \int_r^R p(\rho)d\rho + \mid \mathbf{b}(y) \mid \int_r^R p(\rho)d\rho \\ &\geq \mid c(y) \mid r \left(\int_r^R p(\rho)d\rho - \int_r^R \frac{p(\rho)r}{\rho}d\rho \right) \\ &+ \left| \sum_{i=1}^{3} b_i(y) \frac{y_{(i)} - x_{(i)}}{r} \right| \int_r^R p(\rho)d\rho. \end{aligned} \tag{109}$$

One can see that (108) follows from (109).

Now the representation (95) can be written in the form:

$$u(x) = \int_{B(x)} M_y^* L_p(y,x) u(y) dy + \int_{B(x)} L_p(y,x)\phi(y) dy. \tag{110}$$

The last representation enables the formulation of a unbiased estimate for the solution of the problem under consideration.