A New Iterative Monte Carlo Approach for Inverse Matrix Problem *

I.T. Dimov [1], T.T. Dimov [1] and T.V. Gurov [1]

Abstract

A new approach of iterative Monte Carlo algorithms for the well known inverse matrix problem is presented and studied. The algorithms are based on a special techniques of iteration parameter choice, which allows to control the convergence of the algorithm for any column (row) of the matrix using different relaxation parameters. The choice of these parameters is controlled by a posteriori criteria for every Monte Carlo iteration. The presented Monte Carlo algorithms are implemented on a SUN Sparkstation. Numerical tests are performed for matrices of moderate in order to show how work the algorithms. The algorithms under consideration are well parallelized.

Keywords: Monte Carlo algorithms, iterative methods, Markov chain, inverse matrix problem.

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

1 Introduction

In this work we deal with Monte Carlo algorithms for approximate evaluation of the inverse matrix A^{-1} (A is a square matrix).

Consider the following system of linear equations:

$$Au = b, \tag{1}$$

where

$$A = \{a_{ij}\}_{i,j=1}^{m} \in I\!R^{m\times m}; \quad b, u \in I\!R^{m\times 1}.$$

The inverse matrix problem is equivalent to solve m-times the problem (1), i.e.

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

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

$$AC_j = I_j, \quad j = 1, \ldots, m$$

where

$$I_j \equiv (0, \ldots, 0, \underbrace{1}_{j}, 0, \ldots, 0)^T$$

and

$$C_j \equiv (c_{1j}, c_{2j}, \ldots, c_{mj})^T$$

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

It is also known that Monte Carlo numerical methods give statistical estimates for the solution of a given problem using a certain random variable whose mathematical expectation is the desired solution. These methods are used when not very accurate solution is needed (in the real-life computations the required accuracy is about 1-5 %). The problem of approximate evaluation of the inverse matrices is very useful when one is interested in finding special preconditioning matrices used to accelerate the convergence of basic iterative methods (see, [KY93]).

There are several basic advantages of these algorithms. It is well known that Monte Carlo algorithms are parallel algorithms. They have high parallel efficiency when parallel computers are used [MU49], [DT93], [DT93b]. Monte Carlo algorithms are also very efficient when the problem under consideration is too large or too intricate for other treatment. One of the most important advantages of these algorithms is that they can be used for evaluating only one component of the solution or some linear form of the solution. It is of great practical interest, since the most important problems in the applied sciences are formulated as problems of evaluating linear or non-linear forms of the solution. In this case it is not necessary to perform the all computational work which is needed for obtaining the complete solution. In general there are two classes of Monte Carlo numerical algorithms - direct algorithms and iterative algorithms. The direct algorithms obtain the approximate solution of a problem in a finite number of steps, and contain only a stochastic error.

Iterative Monte Carlo algorithms approximate some deterministic iterative method for a given initial problem. In this case there are two errors - systematic (a truncation error) and stochastic (a probable error). The systematic error depends on the number of iterations of the used iterative method (Section 1), while the stochastic error depends on the probabilistic nature of the Monte Carlo method (Section 3).

It is well known [W68], [C54] that iterative Monte Carlo methods are preferable for solving large sparse systems (such as those arising from approximations of partial differential equations). Such methods are good for diagonally dominant systems in which the rate of the convergence is high.

Consider the linear algebraic system (1). Define an iteration of *order* i as a function of the following form

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

where $u^{(k)}$ is the m-component vector obtained from the k-th iteration. It is desired that

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

The method is called *stationary* if $F_k = F$ for all k, that is, F_k is independent of k. The iterative process is called *linear* if F_k is a linear function of $u^{(k)}, \ldots, u^{(k-i+1)}$. In this paper we study *stationary linear iterative Monte Carlo* algorithms.

Consider the *Jacobi overrelaxation iterative* method with relaxation parameter γ. We shall deal with the matrix $L = \{l_{ij}\}_{ij=1}^{m}$, such that

$$L = I - DA,$$

where D is a diagonal matrix $D = diag(d_1, \ldots, d_m)$ and

$$d_i = \frac{\gamma}{a_{ii}}, \ \ \gamma \in (0,1] \ \ i = 1, \ldots, m.$$

Now, the system (1) can be presented in the following form:

$$u = Lu + f, \tag{2}$$

where

$$f = Db.$$

Let us suppose that the matrix A is diagonally dominant. In fact, this condition is too strong and the presented algorithms work for more general matrices, as it will be shown in Section 4. Obviously, if A is a diagonally dominant matrix, then the elements of the matrix L must satisfy the following condition:

$$\sum_{j=1}^{m} |l_{ij}| < 1 \qquad i = 1, \ldots, m. \tag{3}$$

Consider the *first-order stationary linear iterative process* for the system (2)

$$u^{(k)} = Lu^{(k-1)} + f, \ \ k = 1, 2, \ldots . \tag{4}$$

In fact (4) defines a *Neumann series*

$$u^{(k)} = f + Lf + \ldots + L^{k-1}f + L^k u^{(0)}, \qquad k > 0 \, .$$

From (2) and (4) one can get the value of the truncation error. If $u^{(0)} = f$ then

$$u^{(k)} - u = L^k(f - u).$$

It is well known that property (3) is a sufficient condition for convergence of the Neumann series, i.e.

$$u = \lim_{k \to \infty} u^{(k)}.$$

It is clear that every iterative algorithm uses a finite number of iterations k. In our algorithms we compute the iterations $u^{(q)}$, $1 \leq q \leq k$ using Monte Carlo approach with an additional statistical error. In practice the truncation parameter k is not a priori given parameter. It is obtained from the condition that the difference between the stochastic approximation of two successive approximations is smaller than a given sufficiently small parameter ε.

An important parameter of the algorithmic efficiency is the *computational complexity* or the *time* of the algorithm. We consider some theoretical estimates of the complexity of our algorithms. We also give some numerical results, showing the *computational complexity.* Here we present results for the *computational complexity* of other deterministic or stochastic algorithms. One can find some important estimates in the work of John Halton [Ha92]. It is convenient to present the problem of estimating the inverse matrix as a problem of solving the following system of *linear systems of equations*, of the general form

$$AU = B, \tag{5}$$

where the $(m \times m)$ matrix A and the $(m \times m)$ matrix B are known, while the $(m \times m)$ matrix U is the unknown quantity to be determined.

The inverse matrix problem consists in computing the unknown matrix U in the case when $B = I$. There are many classical numerical methods for solving an $(m \times m \times m)$ system (5) of linear algebraic equations. The *direct methods*, such as the *Gaussian* and *Gauss-Jordan* elimination, take time

$$T_{DIRECT}(m) = O(m^3),$$

while the *iterative methods*, such as the *Jacobi, Gauss-Seidel*, and various *relaxation* techniques, take time

$$T_{ITER}(m, k) = O(m^3 k),$$

if there are k iterations. The direct methods are also well parallelizable, but they need a large number of processors p. It is known that Csanky procedure produces the inverse of a square matrix in $O(log^2 m)$ steps [Csa76], but needs an excessive number of processors $p = m^4$. Note, that the last result is obtained under a very unrealistic condition if one is interested in matrices of large size. In fact, following the principle of decomposition of the algorithm [HK74] one can get the following estimation of the time of the *direct methods* (depending of p)

$$T_{DIRECT}(p, m) = O\left(\frac{m^5}{p} log^2 m\right).$$

It is known that the Monte Carlo techniques ([Ha92]) take time

$$T_{MC}(m, k, n) = O(m^2 kn)$$

(or less), if there are, on average, n samples, involving random walks of average length k, to determine the components of U. In comparison with iterative methods, we have n

replacing m. Thus, so long as $n < m$, this is far more efficient that the classical methods. The presented estimates show that Monte Carlo algorithms are preferable when one need to have a coarse estimation of the inverse matrix. The problem is very important when one is interested in finding *factorized sparse approximate inverse preconditioners* (see, for example [KY93]). If a system of p processors is available, then the Monte Carlo techniques take time

$$T_{MC}(m,k,n,p) = O\left(\frac{m^2}{p}kn\right).$$

Clearly, the rate of convergence (respectively, the average length k of the Markov chain) depends on the spectral radius of the matrix. As long as the spectral radius is smaller, the algorithms under consideration are far more efficient (since k can be a small number for obtaining a good accuracy).

The algorithms considered in this paper have the same rate of complexity as the algorithm described by J. Halton in ([Ha92]), but they are more efficient, because different relaxation parameters and stop criteria are used.

2 Discrete Markov Processes

Here we consider the problem of evaluating the linear form $V(u)$ of the solution u of the system (2):

$$V(u) \equiv (v,u) = \sum_{i=1}^{m} v_i u_i\,, \tag{6}$$

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

We shall construct a random variable $X[v]$, which mathematical expectation is equal to the linear form (6), i.e.

$$EX[v] = V(u).$$

using discrete Markov processes with a finite set of states.

Then the computational problem becomes one of calculating repeated realizations of $X[v]$ and of combining them into an appropriate statistical estimator of $V(u)$. Note that the nature of the every realization of $X[v]$ is a Markov process. We will consider only *discrete Markov processes with a finite set of states*, the so-called *finite discrete Markov chains.*

Definition 2.1 *A finite discrete Markov chain S is defined as a finite set of states $\{s_1, s_2, ..., s_m\}$. At each of a discrete sequence of times $t = 0, 1, \ldots, k, \ldots$ the chain S is in one of the following states $s_{t_0}, s_{t_1}, \ldots, s_{t_k}, \ldots$, which satisfies the Markov property*

$$P(s_{t_q} = \alpha_q | s_{t_{q-1}}, s_{t_{q-2}}, \ldots, s_{t_0}) = P(s_{t_q} = \alpha_q | s_{t_{q-1}}),\;\; q = 0, 1, \ldots,\;\; \alpha_q \in \{s_1, \ldots, s_m\}.$$

Any state s_i is associated with a set of conditional probabilities p_{ij}, such that p_{ij} is the probability that the system, which at the t-th time, is in the state s_i, will be in the state s_j at the $(t+1)$-th time, i.e.

$$P(s_{t_{k+1}} = s_j | s_{t_k} = s_i) = p_{ij}.$$

Thus, p_{ij} is the probability of the transition $s_i \Rightarrow s_j$. The set of all conditional probabilities p_{ij} defines a transition probability matrix $P = \{p_{ij}\}_{i,j=1}^{m}$ which completely determines the probabilities of the given chain S.

Definition 2.2 *The state is called absorbing if the chain terminates in this state with probability one.*

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

$$S_k = s_{t_0} \rightarrow s_{t_1} \rightarrow s_{t_2} \rightarrow \ldots \rightarrow s_{t_k}, \tag{7}$$

where s_{t_q}, $(q = 1, \ldots, k)$ is one of the absorbing states. This determines the value of some function $F(S) = X[v]$, which depends on the sequence (7). The function $F(S)$ is a random variable. After the value of $F(S)$ has been calculated, the system is restarted to its initial state s_{t_0} and the transitions are begun anew. A number of n independent runs are made through the Markov chain starting from the state s_{t_0} to any of the absorbing states. The average

$$\frac{1}{n} \sum_{T} F(S) \tag{8}$$

is taken over all actual sequences of transitions (7). The value in (8) approximates $E\{F(S)\}$, which is the required linear form of the solution.

We also will be interested in *computational complexity.*

Definition 2.3 *The computational complexity is defined by*

$$nE(k)t_0,$$

where $E(k)$ is the mathematical expectation of the number of transitions in the sequence (7) and t_0 is the mean time needed for realization of one transition.

In fact, the definition of the computational complexity is used for obtaining theoretical estimates, because one can only estimate the mathematical expectation of the number of transitions k. In practice for every realization of a given Monte Carlo algorithm one has a determined number of moves in all realizations of the Markov chain which we denote by R:

$$R = \sum_{i=1}^{n} k_i, \tag{9}$$

where k_i is the number of moves of the ith realization of the Markov chain.

3 Iterative Monte Carlo Method

Consider the Markov chain

$$S = s_0 \to s_1 \to s_2 \to \ldots \to s_k \to \ldots \tag{10}$$

with m states $\{1, 2, \ldots, m\}$. Let

$$P(s_0 = i) = p_i, \quad P(s_q = j | s_{q-1} = i) = p_{ij},$$

where p_i (initial probability) is the probability that the chain starts in state i and p_{ij} is the transition probability for the random process to go to state j after being in state i. Probabilities p_{ij} define a transition matrix $P = \{p_{ij}\}_{ij=1}^{m}$. Clearly p_i and p_{ij} must be nonnegative and

$$\sum_{i=1}^{m} p_i = 1, \quad \sum_{j=1}^{m} p_{ij} = 1, \quad \text{for any} \quad i = 1, 2, ..., m.$$

Define the weight function W_q, for Markov chain (10) with m states, using recursion formula

$$W_0 = 1; \qquad W_q = W_{q-1} \frac{l_{s_{q-1} s_q}}{p_{s_{q-1} s_q}}, \qquad q = 1, 2, \ldots$$

where the sequence of states $s_0, s_1, s_2, \ldots$ is a given trajectory with initial probability p_{s_0} and transition probabilities $p_{s_{q-1} s_q}$.

In fact

$$W_q = \frac{l_{s_0 s_1} l_{s_1 s_2} \ldots l_{s_{q-1} s_q}}{p_{s_0 s_1} p_{s_1 s_2} \ldots p_{s_{q-1} s_q}}. \tag{11}$$

Define the following random variable

$$X[v] = \frac{v_{s_0}}{p_{s_0}} \sum_{q=0}^{\infty} W_q f_{s_q}. \tag{12}$$

The following statement holds.

Theorem 3.1 *Let u be a solution of the system (2) and the elements l_{ij} of the matrix L satisfy the property (3). Then the mathematical expectation of the random variable $X[v]$ is equal to the linear form $V(u)$ defined by (6), i.e.*

$$EX[v] = (v, u).$$

Proof. The proof is similar to the proof of J. Spanier and E.M. Gelbard (see, [SG69]). In fact, for a given sequence of states $\alpha_0 \to \alpha_1 \to \cdots \to \alpha_q \to \cdots$ we consider the Markov chain, defined by (10).

Then we have

$$P\{s_0 = \alpha_0, \cdots, s_q = \alpha_q\} = p_{\alpha_0} p_{\alpha_0 \alpha_1} \cdots p_{\alpha_{q-1} \alpha_q},$$

where

$$\alpha_i \in \{1, 2, \ldots, m\} \quad i = 1, 2, \ldots, q.$$

Using (11) one can obtain

$$E\{\frac{v_{s_0}}{p_{s_0}} W_q f_{s_q}\} = \sum_{\alpha_0, \cdots, \alpha_q = 1}^{m} \frac{v_{\alpha_0}}{p_{\alpha_0}} \frac{l_{\alpha_0\alpha_1} l_{\alpha_1\alpha_2} \ldots l_{\alpha_{q-1}\alpha_q}}{p_{\alpha_0\alpha_1} p_{\alpha_1\alpha_2} \ldots p_{\alpha_{q-1}\alpha_q}} f_{\alpha_q} p_{\alpha_0} p_{\alpha_0\alpha_1} p_{\alpha_1\alpha_2} \ldots p_{\alpha_{q-1}\alpha_q}$$

$$= \sum_{\alpha_0=1}^{m} v_{\alpha_0} \sum_{\alpha_1=1}^{m} \cdots \sum_{\alpha_{q-1}=1}^{m} l_{\alpha_0\alpha_1} l_{\alpha_1\alpha_2} \ldots l_{\alpha_{q-2}\alpha_{q-1}} \sum_{\alpha_q=1}^{m} l_{\alpha_{q-1}\alpha_q} f_{\alpha_q}$$

$$= \sum_{\alpha_0=1}^{m} v_{\alpha_0} (L^q f)_{\alpha_0} = (v, L^q f).$$

Therefore

$$EX[v] = \sum_{q=0}^{\infty} E\{\frac{v_{s_0}}{p_{s_0}} W_q f_{s_q}\}$$

$$= \sum_{q=0}^{\infty} (v, L^q f) = (v, u).$$

This completes the proof. $\diamondsuit$

Obviously if $u^{(k)}$ is the k-th iterative solution of (4) with $u^{(0)} = f$ then the mathematical expectation of the random variable

$$X_k[v] = \frac{v_{s_0}}{p_{s_0}} \sum_{q=0}^{k} W_q f_{s_q} \tag{13}$$

is equal to the linear form $V(u^{(k)}) = (v, u^{(k)})$, i.e.

$$EX_k[v] = (v, u^{(k)}).$$

The parameter k is chosen from the following condition of truncation of the Markov chain

$$|W_q| < \varepsilon. \tag{14}$$

As an approximate value of the linear form (6) we get the following average of the random variable $X[v]$

$$\overline{X}_n[v] = \frac{1}{n} \sum_{i=1}^{n} \{X[v]\}_i,$$

where $\{X[v]\}_i$ is the i-th independent realization of the random variable (12). The probable error is defined as a value r_n (see, for example [So73], [EM82]) for which the following condition

$$P\{|\overline{X}_n - EX[v]| < r_n\} \approx \frac{1}{2} \approx P\{|\overline{X}_n - EX[v]| > r_n\}.$$

is fulfilled.

It is easy to show (following [So73]) that for algorithms under consideration

$$r_n \approx 0.6745\sqrt{VarX[v]/n},$$

where

$$VarX = E(X^2) - (EX)^2.$$

A good choice of the initial probability vector $p = \{p_i\}_{i=1}^m$, namely

$$p_i = \frac{|v_i|}{\sum_{j=1}^m |v_j|},$$

and the transition probability matrix $P = \{p_{ij}\}_{i,j=1}^m$, namely

$$p_{ij} = \frac{|l_{ij}|}{\sum_{j=1}^m |l_{ij}|} \quad i = 1, 2, \ldots, m$$

leads to the so-called *Monte* Carlo *Almost Optimal* (MAO) algorithm (see the definition of the MAO algorithm in [MAD94]).

Remark 1.

It is clear that if we choose the special case of linear form $V(u)$ according to vector $v = I_{j_0} = (0, \ldots, 0, \underbrace{1}_{j_0}, 0, \ldots, 0)^T$, (where the one is in the "j_0" place) than we get "j_0" component of the solution.

Remark 2.

Both errors - truncation and probable arise when the random variable (12) is replaced by random variable (13). As an approximate value of the linear form (6) we get the mean value of the random variable (13).

4 Iterative Monte Carlo Algorithms

In our iterative Monte Carlo algorithms we use the following norms of vectors $u \in I\!R^m$ and matrices $A \in I\!R^{m\times m}$ [Ax94]:

$$\| u \|_2 = \left(\sum_{i=1}^m |u_i|^2 \right)^{1/2} \quad \text{(the Euclidean norm)} \tag{15}$$

$$\| u \|_\infty = \max_i |u_i| \quad \text{(the maximum norm)} \tag{16}$$

$$\| A \|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{m} a_{ij}^2 \right)^{1/2} \quad \text{(the Frobenius norm)} \tag{17}$$

The first algorithm is auxiliary and evaluates every component of the solution u of the linear algebraic system (1).

Algorithm 4.1 *:*

```
1. Input initial data: the matrix $A$, the vector $b$, the constant $\varepsilon$, the relaxation parameter $\gamma \in (0,1]$ and the integer $n$.
2. Preliminary calculations (preprocessing):
    2.1. Compute the matrix $L$:
        $\{l_{ij}\}_{i,j=1}^{m} = \begin{cases} 1-\gamma & \text{when} \quad i = j \\ -\gamma \frac{a_{ij}}{a_{ii}} & \text{when} \quad i \neq j \, ; \end{cases}$
    2.2. Compute the vector $f$:
        $f_i = \gamma \dfrac{b_i}{a_{ii}}, \quad i = 1, \ldots, m;$
    2.3. Compute the vector $lsum$:
        $lsum(i) = \sum_{j=1}^{m} |l_{ij}| \quad for \quad i = 1, 2, \ldots, m;$
3. For $i_0 := 1$ to $m$ do step 3.1 and step 3.2.
3.1. Perform one trajectory:
    3.1.1. Set initial values $X := 0$ , $W := 1$ ;
    3.1.2. Calculate $X := X + W f_{i_0}$ ;
    3.1.3. Generate an uniformly distributed random number $\xi \in (0,1)$;
    3.1.4. Set $j := 1$;
    3.1.5. If $(\xi < \sum_{k=1}^{j} p_{i_0 k})$ then
        3.1.5.1. Calculate $W := W \times sign(l_{i_0 j}) \times lsum(i_0)$ ;
        3.1.5.2. Calculate $X := X + W f_j$ ( one move in trajectory );
        3.1.5.3. If $|W| < \varepsilon$ then go to step 3.1.6 ;
        3.1.5.4. Update the index $i_0 := j$ and go to step 3.1.3;
                 else
        3.1.5.5. Update $j := j + 1$ and go to step 3.1.5.
    3.1.6. End of the trajectory.

3.2. Calculate the mean value based on $n$ independent trajectories:
    3.2.1. Do $n$ times step 3.1;
    3.2.2. Calculate $\overline{X}_n$ and $u_{i_0} := \overline{X}_n$.

4. End of the Algorithm 4.1.
```

The Algorithm 4.1 describes the evaluation of every component of the solution u of the problem (1), which is, in fact, a linear algebraic system. Algorithm 4.1 is considered separately, since it (or some of its steps) will be used in next algorithms.

For finding the corresponding j-th component of the solution the following linear form is used

$$V(u) = (v, u),$$

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

The second algorithm computes the approximation $\hat{C}$ to the inverse matrix $C = A^{-1}$. The algorithm is based on special techniques of relaxation parameter choice. The choice of the relaxation parameter γ can be controlled by a posteriori criteria for each column of the residual matrix $E = A\hat{C} - I$. Each column of the matrix $\hat{C}$ is computed independently using Algorithm 4.1.

Algorithm 4.2 *:*

```
1. Input initial data: the matrix $A$, the constant $\varepsilon$, the parameter eucl_norm (a sufficiently large given value), the integers $n$ and $l$, and the set of the relaxation parameters $\{\gamma_1, \gamma_2, \ldots, \gamma_l\}$, where $\gamma_i \in (0, 1]$, $i = 1, \ldots, l$.

2. For $j_0 := 1$ to $m$ do

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

    2.1. For $(k := 1)$ to $l$ do
    2.1.1. Apply the Algorithm 4.1 without step 2.2 for $\varepsilon$, $n$, $\gamma = \gamma_k$, and the right-hand side vector $f = I_{j_0} = (0, \ldots, 0, \underbrace{1}_{j_0}, 0, \ldots, 0)^T$ to obtain the column–vector
        $\hat{C}^{(k)}_{j_0} = (\hat{c}^{(k)}_{1j_0}, \ldots, \hat{c}^{(k)}_{mj_0})^T$;
        2.1.2. Compute the Euclidean norm of the residual column–vector $E_{j_0}^{(k)}$:
               $\| E_{j_0}^{(k)} \|_2 = \left( \sum_{j=1}^{m} \left( \sum_{i=1}^{m} a_{ji} \hat{c}_{ij_0}^{(k)} - \delta_{jj_0} \right)^2 \right)^{1/2}$ ;
        2.1.3. If ($\| E_{j_0}^{(k)} \|_2 < eucl\_norm$ then
               $\hat{C}_{j_0} := \hat{C}_{j_0}^{(k)}$;
               $eucl\_norm := \| E_{j_0}^{(k)} \|_2$.
  3. End of the Algorithm 4.2.
```

Algorithm 4.2 is based on Algorithm 4.1 finding different column–vectors of the matrix $\hat{C} = (\hat{C}_1, \ldots, \hat{C}_m)$ by using corresponding values of the relaxation parameter $\gamma = \gamma_p,\ \ p = 1, 2, \ldots, l$. The values of γ_p are chosen such that to minimize the Euclidean norm of the following column–vectors:

$$E_j = A\hat{C}_j - I_j, \quad j = 1, 2, \ldots, m,$$

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

The use of the criterion of minimization of the Euclidean norm (15) of the column–vector E_j over the set γ permits to find better approximation of the residual matrix $E = (E_1, \ldots, E_m)$ and hence improves the approximation of the inverse matrix $\hat{C}$ column by column.

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

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

The third algorithm essential require the Monte Carlo approach and there is not deterministic analogy.

Algorithm 4.3 *:*

```
  1. Input initial data: the matrix $A$, the integers $n$ and $l$, the sets of parameters
     $\{\varepsilon_1, \ldots, \varepsilon_m\}$, $\{\gamma_1, \gamma_2, \ldots, \gamma_l\}$, ($\gamma_i \in (0,1]$, $i = 1, \ldots, l$) and the sufficiently large
     values $\{max\_norm_1, \ldots, max\_norm_m\}$.
  2. For $k := 1$ to $l$ do
        2.1. Apply step 2 of Algorithm 4.1 for $\gamma := \gamma_k$ without step 2.2;
        2.2. For $i_0 := 1$ to $m$ do
        2.3. For $j_0 := 1$ to $m$ do
        2.3.1. Apply step 3.1 and step 3.2 of the Algorithm 4.1 for $n, \gamma = \gamma_k$ , $\varepsilon := \varepsilon_{i_0}$
               and the right-hand side vector $f = I_{j_0} := (0, \dots, 0, \underbrace{1}_{j_0}, 0, \dots, 0)$ to compute
               the approximate inverse matrix $\hat{C}^{(k)} = \{\hat{c}^{(k)}_{i_0 j_0}\}^{m}_{i_0 j_0 = 1}$.
    2.4. For $i_0 := 1$ to $m$ do
        2.4.1. Calculate the maximum norm of the residual row–vector $E^{(k)}_{i_0}$:
                   $\parallel E^{(k)}_{i_0} \parallel_\infty = \max_{i \in \{1,2,\dots,m\}} |\sum_{j=1}^{m} \hat{c}^{(k)}_{i_0 j} a_{ji} - \delta_{i_0 i}|$;
        2.4.2. If $(\parallel E^{(k)}_{i_0} \parallel_\infty < max\_norm_{i_0})$ then
                   $\hat{C}_{i_0} := \hat{C}^{(k)}_{i_0}$;
                   $max\_norm_{i_0} := \parallel E^{(k)}_{i_0} \parallel_\infty$.
3. End of the Algorithm 4.3.
```

The difference between the last two algorithms is that the Algorithm 4.3 cannot be applied in traditional (non-stochastic) iterative methods. The traditional methods allow to evaluate the columns of the inverse matrix in parallel, but they do not allow to obtain their elements independently of 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 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 moduli of the non-diagonal entries to the value of the diagonal element).

One has to mention that the computational complexity of Algorithm 4.3 also depends on "how ill-conditioned" is a given row of the matrix A. The given row A_i of the matrix A is "ill-conditioned", when the property

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

of *strictly diagonally dominant matrices* is not satisfied (but all the eigenvalues lie inside of the unit circle).

The Algorithm 4.3 presented above is very convenient for such matrices since it chooses the value of the relaxation 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:

$$b_i = \sum_{j=1}^{i-1} |a_{ij}| + \sum_{j=i+1}^{m} |a_{ij}| - |a_{ii}|. \qquad (18)$$

The possibility to treat matrices which are not strictly diagonally dominant increases the set of the problems treated using Monte Carlo algorithms. For finding different row–vectors of the approximation of the inverse matrix $\hat{C} = (\hat{C}_1, \dots, \hat{C}_m)^T$ different number of moves (iterations) can be used. The number of moves are controlled by parameters ε_i $(i = 1, \ldots, m)$ from the inequality (14). 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. When $b_i > 0$ is larger, then the convergence is low, and thus the difference between two iteration (controlled by ε) has to be smaller. The procedure, used a different ε we call *fine stop criterion.* 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. This allows to keep the parameter R (9) witch determine the computational complexity on the same level. For an a posteriori criterion we use the minimization of the maximum norm (16) of the following row–vectors

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

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

The use of the criterion of minimization of the maximum norm of the row–vectors E_i improves the characteristic of the residual matrix $E = (E_1, \ldots, E_m)^T$:

$$E = \hat{C} A - I. \tag{19}$$

and hence leads to better approximation of the matrix $\hat{C}$ row by row.

One can also control the number of moves in the Markov chain (that is the number of iterations) such that to have a good balance between the stochastic and systematic error (i.e., the truncation error). The problem of balancing of both - 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_k, that is

$$r_n = O(r_k).$$

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 $E(k)$ of the number of steps in each random trajectory. The balancing allows to increase the accuracy of the algorithm for a fixed computational complexity, because in this case one can control the parameter R, defined in (9), by choosing different lengths of the realizations of the Markov chain. In practice, we choose the absorbing state of the random trajectory using the inequality (14).

5 Discussion of the Numerical Results

As an example we consider matrices arising after application of the mixed finite element method for the following boundary value problem

$$\left| \begin{array}{l} -\mathrm{div}(A_c \underline{\nabla} p)(x) = f(x), \text{ in } \Omega \\ p = 0, \text{ on } \partial\Omega, \end{array} \right. \tag{20}$$

where Ω is a rectangular subdomain of $I\!R^2$ and $A_c(x)$ is a diagonal matrix which 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

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

respectively.

Then the mixed variational formulation of the problem (20) is given by characterizing the pair $(\underline{u}, p)$, as the solution of

$$\left| \begin{array}{ll} a(\underline{u}, \underline{v}) + b(\underline{v}, p) \;=\; 0, & \forall \underline{v} \in \underline{V}; \\ b(\underline{u}, w) \;=\; -(f, w), & \forall w \in W, \end{array} \right. \tag{21}$$

where

$$a(\underline{u}, \underline{v}) \;=\; (\alpha_1 u_1, v_1) + (\alpha_2 u_2, v_2), \quad b(\underline{u}, w) \;=\; (div\underline{u}, w)$$

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

The mixed finite element approximation of problem (21) with rectangular Raviart-Thomas elements leads to the following linear algebraic system [AD94], [RT91]:

$$Ku = \begin{pmatrix} A_1 & 0 & B_1 \\ 0 & A_2 & B_2 \\ B_1^T & B_2^T & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ p \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -f \end{pmatrix}, \tag{22}$$

where A_i are $m \times m$ matrices, B_i are $m \times m_1$ matrices $(m_1 \leq m)$, $u_i \in I\!R^m$ and $p, f \in I\!R^{m_1}$, $\;i = 1, 2$.

If $A_i^{-1}\,(i = 1, 2)$ is obtained then the system (22) 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 two m-dimensional systems and one m_1-dimensional system.

Column	Euclidean norm				Maximum 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

Table 1: Connection between ε and γ. Here $m = 16$, $n = 24$.

The matrices $A = A_i,\ i = 1, 2$ obtained from the lowest order mixed finite element approximation on rectangular grid are strictly diagonally dominant. For the above mentioned matrices the Algorithm 4.2 is applied. Numerical examples for a matrix $A \in I\!R^{16\times 16}$, for different values of the relaxation parameter γ are presented.

The values $\gamma_i = i/10\ \ (i = 1, \dots, 10)$ of the parameter γ for different columns of the matrix $\hat{C}$ are shown in Table 1.

In general, matrices arising after the mixed finite element approximation are not strictly diagonally dominant, but the eigenvalues of the first order mixed finite element discretization lie in the unit circle. As a basic test example for applying Algorithm 4.3 a matrix $A \in I\!R^{7\times 7}$, which rows have a typical properties of these matrices, is used. In this case parameters b_i (18) are:

$$(b_1, b_2, b_3, b_4, b_5, b_6, b_7) = (-14, -10, -6, 6, -6, -10, -14)$$

The size of the matrix is relatively small, because we just want to demonstrate how works the Algorithm 4.3. Here we also have to mention that the computational complexity of the algorithm practically does not depend on the size of the matrix. In fact [DK96] the computational complexity of our algorithms depends linearly on the mean value of the number of non-zero entries 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 compute the Frobenius norm (17) of the residual matrix.

Some of the numerical results are shown on Figures 1 – 8 and by Table 2. On all Figures the value of the Frobenius norm of the residual matrix E is denoted by F.N., the number

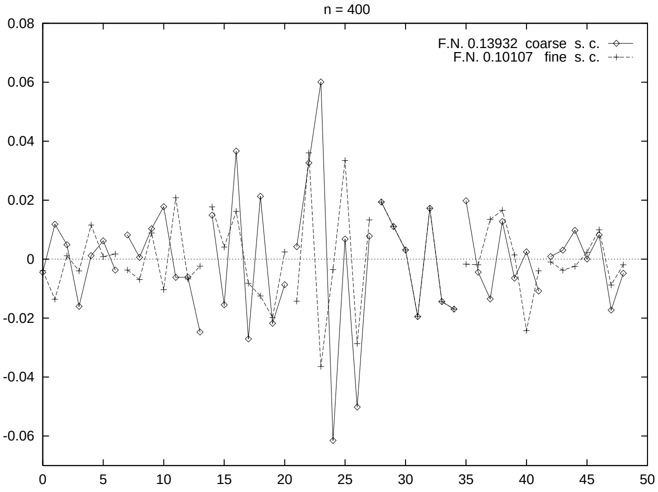

Figure 1
Non-balanced case

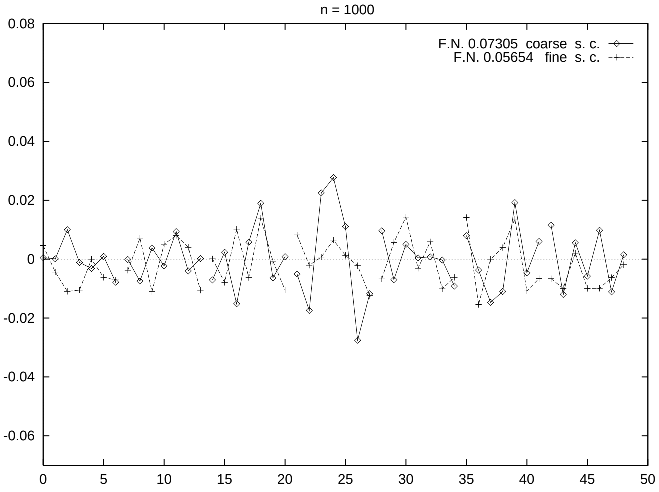

Figure 2
Balanced case

of realizations of the random variable (i.e., the number of random trajectories) is denoted by n and the value *epsilon* is the parameter used in the stop-criterion (14).

Figure 1 presents the values of the residual matrix (19) in both cases under consideration – coarse stop criterion (' $\diamondsuit$ ') and fine stop criterion ('+'). The first set of connected points corresponds to values of the first row of the residual matrix, the second set – to the second row of the same matrix, etc. When the coarse stop criterion is used $\varepsilon = 0.0001$. When the fine stop criterion is used, different values $\varepsilon_1, \ldots, \varepsilon_7$ of ε are applied such that the computational complexity is smaller than in comparison with the case if the coarse stop criterion (see, also Table 2). 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 1 correspond to the non-balanced case.

Similar results when $n = 1000$ and $\varepsilon = 0.001$ (for the coarse stop criterion) are presented on Figure 2. One can see, that

- ε is 10 times larger than in the previous case, but the Frobenius norm is about twice as small, because the number of realizations is larger.

The results presented on Figure 1 and Figure 2 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 it is to have a good balancing between the stochastic and systematic error. The computational effort for the cases presented on Figure 1 and Figure 2 is approximately equal, but the results in the case of Figure 2, when we have a good balancing are almost twice as good.

Let us discuss the result presented on Figures 3 and 4. Here instead of elements of the residual matrix E, the maximum norm of the every its row $\| E_j \|_\infty$, $j = 1, \ldots, m$ (16) are shown. If the computational complexity for a constant ε is denoted by R_c and the

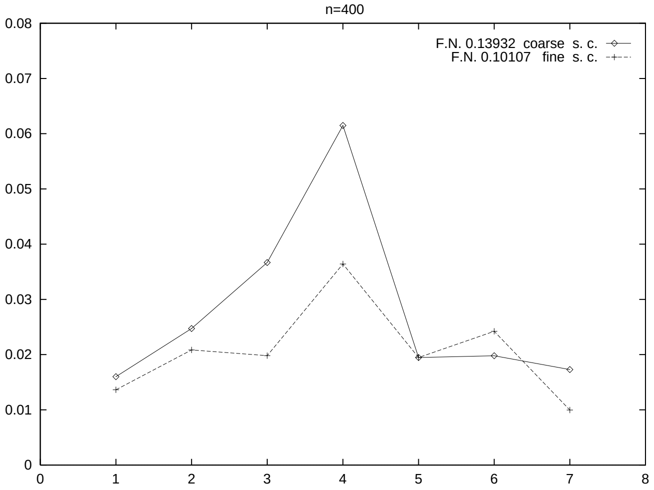

Figure 3
Non-controlled balance

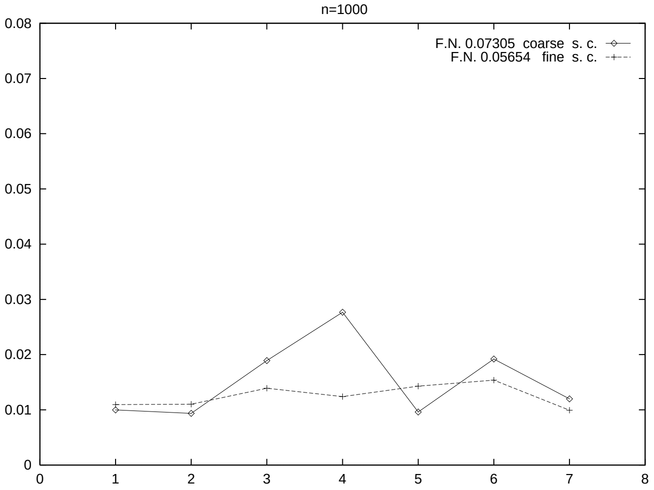

Figure 4
Controlled balance

computational complexity when different values of $\varepsilon = \varepsilon_i,\ i = 1, \ldots, 7$ is denoted by R_f we consider the case when

$$\frac{R_c}{R_f} \geq 1.$$

The results presented on Figures 3 and 4 show that apart from the less computational complexity R_f of the fine stop criterion algorithm it gives better results than the coarse stop criterion algorithm with complexity R_c (see Table 2). This fact is observed in both cases - balanced (Figure 3) and non-balanced (Figure 4).

- 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 5 and 6 present test results for the modulus of every element of the residual matrix (19) when the coarse stop criterion and fine stop criterion respectively are used in the non-balanced case. The indices of the rows and columns of the matrix are shown on axes X and Y respectively.

One can see, that

- the Frobenius norm of the estimate in the case of fine stop criterion is about 1.4 times smaller than the corresponding value for the coarse stop criterion, and
- the variances of the estimate of the case of fine stop criterion are smaller.

Figures 7 and 8 show results similar to those of Figures 5 and 6 in the balanced case. One can make the same conclusion as in the non balanced case, but here

- the Frobenius norm is almost twice as small.

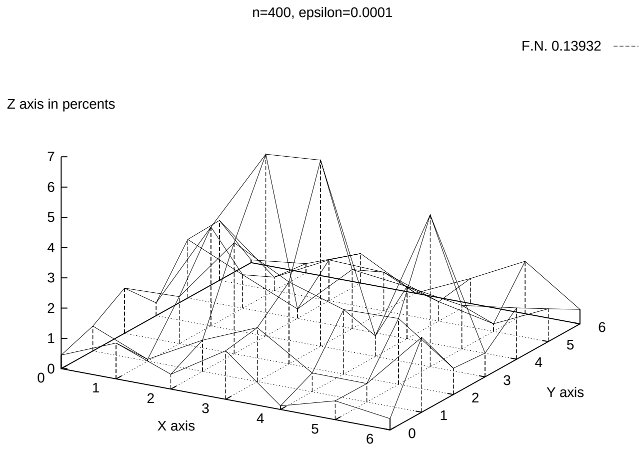

Figure 5
Coarse stop-criterion

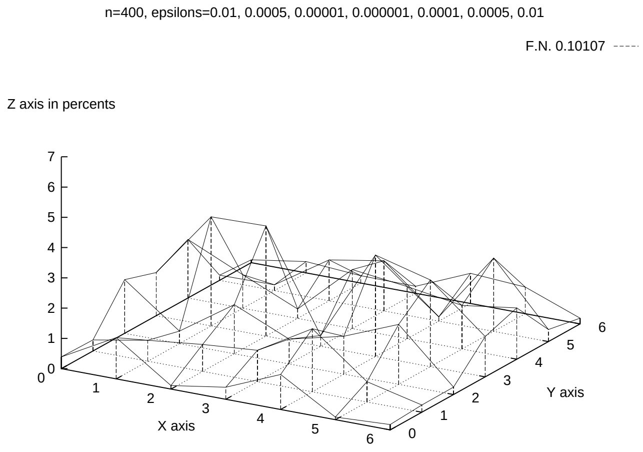

Figure 6
Use of different fine stop-criterion

γ	Not balanced case		Balanced case	
	Coarse s. c. R_c/R_f	Fine s. c. R_f/R_f	Coarse s. c. R_c/R_f	Fine s. c. R_f/R_f
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
Average value	1.0848	1	1.0358	1

Table 2: Computational complexity

6 Conclusion

Iterative Monte Carlo algorithms are presented and studied. These algorithms can be used for solving inverse matrix problems.

The following conclusion can be drawn:

- 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 improve the algorithm using error estimate criterion "column by

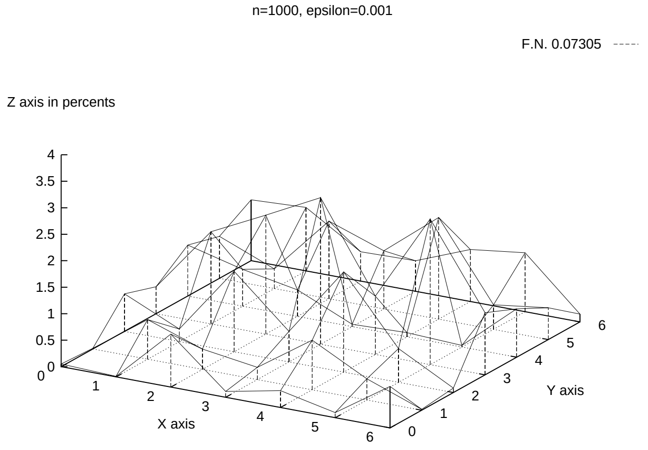

Figure 7
Controlled balancing – coarse stop criterion

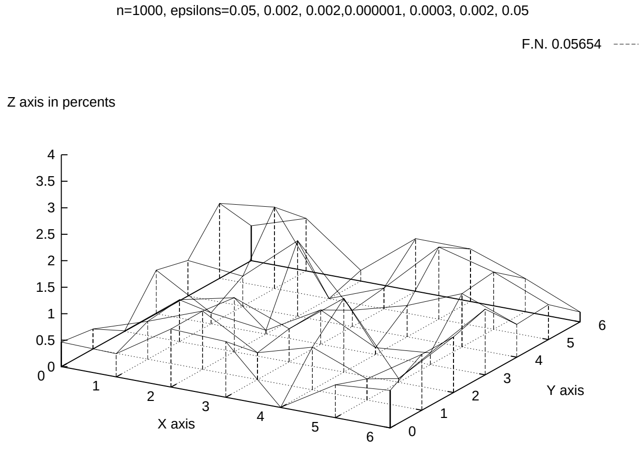

Figure 8
Controlled balancing – fine stop criterion

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 efficiently implemented on MIMD-machines.

References

[AD94] A. Andreev, T. Dimov, *Lumped Mass Approximation for the Mixed Finite Element Method*, in: **Advances in Numerical Methods and Applications -** $O(h^3)$, Proceedings of the Third International Conference on Numerical Methods and Applications, Sofia, 21-26 August 1994, World Scientific, Singapore, New Jersey, London, Hong Kong, pp. 11–18.

[Ax94] O. Axelsson, *Iterative Solution Methods*, **Cambridge University Press**, Cambridge, 1994.

[Csa76] L. Csanky, *Fast parallel matrix inversion algorithms*, **SIAM J. Comput. vol. 5**, (1976), No 4, pp. 618-623.

[C54] Curtiss J.H., *Monte Carlo methods for the iteration of linear operators*, **J. Math Phys., vol. 32** (1954), No 4, pp. 209–232.

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

[DT93b] I. Dimov, O. Tonev, *Monte Carlo algorithms: performance analysis for some computer architectures.* **J. Computational and Applied Mathematics, vol. 48** (1993) pp. 253–277.

[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.

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

[Ha92] J.H. Halton, *Sequential Monte Carlo Techniques for the Solution of Linear Systems*, **TR 92-033**, University of North Carolina at Chapel Hill, Department of Computer Science, 46 pp., 1992.

[HK74] Hyafil L., H.T. Kung, *Parallel algorithms for solving triangular linear systems with small parallelism*, Dept. of Comp. Sci., Carnegie- Mellon Univ., Dec., 1974.

[KY93] L.Yu. Kolotilina, A. Yu. Yeremin, *Factorized Sparse Approximate Inverse Preconditionings I. Theory*, **SIAM J. Matrix Anal. Appl., vol. 14** (1993), No 1, pp. 45–58.

[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. Statist. Assoc., 44**, (1949), pp. 335–341.

[RT91] J.E. Roberts, J.M. Thomas, *Mixed and Hybrid Methods*, in P.G. Ciarlet and J.L. Lions, Eds, **Handbook of Numerical Analysis, Vol. 2, North-Holland**, Amsterdam, 1991, pp. 523-639.

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

[SG69] J. Spanier, E.M. Gelberd, *Monte Carlo Principles and Neutron Transport problems*, **Addison-Wesley Publishing Company, Reading, Ma**, 1969.

[W68] J.R. Westlake, *A Handbook of Numerical matrix Inversion and Solution of Linear Equations*, **John Wiley & Sons, inc.**, New York, London, Sydney, 1968.