A New Highly Convergent Monte Carlo Method for Matrix Computations *

I.T. Dimov [1], V.N. Alexandrov [2]

Abstract

In this paper a second degree iterative Monte Carlo method for solving Systems of Linear Algebraic Equations and Matrix Inversion is presented. Comparisons are made with iterative Monte Carlo methods with degree one. It is shown that the mean value of the number of chains N, and the chain length T, required to reach given precision can be reduced. The following estimate on N is obtained $N = N_c/\left(c_N + bN_c^{1/2}\right)^2$, where N_c is the number of chains in the usual degree one method. In addition it is shown that $b > 0$ and that $N < N_c/c_N^2$. This result shows that for our method the number of realizations N can be at least c_N^2 times less than the number of realizations N_c of the existing Monte Carlo method.

For parallel implementation, i.e. regular arrays or MIMD distributed memory architectures, these results imply faster algorithms and the reduction of the size of the arrays. This leads also in applying such methods to the problems with smaller sizes, since until now Monte Carlo methods are applicable for large scale problems and when the component of the solution vector or element or row of the inverse matrix has to be found.

Key words: Monte Carlo method, matrix computations, convergence, eigenvalues, convergent iterative process, parallel computation

MSC subject classification: 65 C 05, 65 U 05

*Supported by **The Royal Society (UK)**, partially supported by the **NATO** Grant R406 / 02640 and by the **Ministry of Science and Education of Bulgaria** under Grants # I 501 and # MM 448.

[1]Center for Informatics and Computer Technology, Bulgarian Academy of Sciences, Acad. G. Bonchev Str.,bl. 25 A Sofia, 1113, Bulgaria

[2]Department of Computer Science, University of Liverpool, Liverpool,UK

1 Introduction

The problem under consideration is that of inverting a matrix A or solving systems of linear algebraic equations of the form

$$Au = b, I\!R$$

by the Monte Carlo method, where

A is a square $n \times n$ matrix;

$u = (u_1, u_2, \ldots, u_n)^T$ is a $n \times 1$ solution vector, and

$b = (b_1, b_2, \ldots, b_n)^T$ is a given vector.

Clearly the two problems are equivalent because given A^{-1}, the solution to $Au = b$ is $u = A^{-1}b$. Normally the direct construction of A^{-1} is avoided because of ill-conditioning

It is known that Monte Carlo methods give statistical estimates for the elements of the inverse of the matrix or for components of the solution vector of a linear system by performing random sampling of a certain chance variable whose mathematical expectation is the desired solution [HH64], [So73], [W68]. Moreover, the same algorithmic cost is needed for estimating a linear functional of the solution vector (for example, an inner product of a given vector with the solution vector of the linear algebraic system) [DT90], [DT93b] .

In general Monte Carlo numerical methods may be divided into two classes- **direct methods** and **iterative** ones. The direct methods estimate the solution of the equation in a finite number of steps, and contain only a stochastic error. A direct Monte Carlo method is, for example, the interpolation Monte Carlo method for evaluating integrals [HH64], [So73]. Iterative Monte Carlo methods begin with an approximate solution and obtain an improved solution with each step of the method. In principle they require an infinite number of steps to obtain an exact solution, but usually we will be happy with an approximation to say t significant figures for which there are two types of errors - **stochastic** and **systematic**. The systematic error depends on the number of iterations performed, and of the characteristic values of the matrix, while the stochastic errors depend on the probabilistic nature of the method.

Iterative methods are preferred for solving large sparse systems (such as those arising from approximations of partial differential equations). Such methods are good for dominant-diagonal systems in which convergence is rapid. They are not so useful for dense matrices.

To place the discussion in the correct mathematical framework, define an iteration of degree j as a function of the form

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

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

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

Usually the degree 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 linear if F_k is a linear function of $u^k, \ldots, u^{(k-j+1)}$. The systematic error is the most interesting to us. We will analyse the systematic error by using a special mapping procedure.

Monte Carlo methods are very efficient when parallel processors or parallel computers are available, because the methods are inherently parallel and have loose dependencies. Indeed, in the first paper on the Monte Carlo method [MU49] it was noted that "the statistical methods can be applied by many computers working in parallel and independently".

We concentrate on the Monte Carlo method applied to matrix inversion (MI) and system of linear algebraic equations (SLAE), since only $O(NT)$ steps are required to find an element of the inverse matrix or component of the solution vector, where N the mathematical expectation of the number of chains and T a measure on the chain length in the stochastic process are independent of n. Thus, for sufficiently large n the Monte Carlo approach is theoretically more efficient [C56]. Note that the direct methods of solution require $O(n^3)$ sequential steps (for dense matrices) when some elimination or annihilation scheme (e.g non-pivoting Gaussian Elimination, Gauss-Jordan methods) are employed [BT88]. As a result, the computation time for large problems or real-time problems is prohibitive and prevents the use of many established algorithms and results.

There are still very few results in the area of parallel Monte Carlo methods for Linear Algebra problems. Some regular arrays for matrix inversion and finding a solution vector of system of linear algebraic equations by Monte Carlo method have been recently derived by Megson, Alexandrov, Dimov [MAD93, MAD94a, AM96]. Regular arrays for matrix inversion that have been derived from affine and uniform recurrences using linear timing schedules have received considerable attention in the literature [D88, EA89, KLL87, M92]. Most of the methods are based on variations of the Gauss-Jordan algorithm, or triangularization with the assumption that the matrix being inverted is diagonally dominant or positive definite so that no pivoting is required. If no data is duplicated in the array then approximately $5n$ steps and $O(n^2)$ processors are required for matrix inversion. If data duplication is allowed, then $4n$ steps and $O(n^2)$ processors are possible. In [MAD93] the authors demonstrated the existence of an array which required $O(n^4)$ cells and a time bounded above by $4n$: in fact, an array with $O(n^2NT)$ cells and time $3n/2 + N + T$ was constructed. In [MAD94a] these results were extended to derive a fixed size array independent of n: this array requires $m_2 + 2m_1 + T + N - 1$ time steps and $(m_2 + 2)TN$ cells for a single row of the inverse where $m_1 m_2 = n^2$, N is the mathematical expectation of the number of chains required to reach given precision, and T is the mathematical expectation of chain length in the stochastic process and n is the matrix order. When $m_2 = m$ and $m_1 = n^2/m$ for $m > 0$ a constant, the matrix can be restructured to produce a run-time of $n^2/m + 2m + T + N - 1$ and $(m+2)TN$ cells. A simple re-timing produces a multi-pass organisation requiring only $(m + 2)T$ cells with the effective length of $m_2 = Nn^2/m$.

Parallel Monte Carlo methods with reduced probable error for MI running on cluster of workstations under PVM have been developed by Alexandrov and Lakka [AL96]. Some comparison of parallel Monte Carlo methods for MI can be found in [L95]. The algorithms have very high efficiency, above 0.9.

In this paper we continue the work in [MAD93] by exploiting Monte Carlo methods with minimum probable error. We introduce a new mathematical framework, that develops an alternative Monte Carlo method, which produces a reduction in N and T making it applicable to smaller size matrices. In particular, we present a new iterative Monte Carlo method with high convergence rate. In comparison with existing Monte Carlo methods (which are stationary iterative methods of degree $j = 1$) the new one has degree $j = 2$.

This method allows a smaller number of iterations (respectively, smaller T) for reaching a given error. In addition, when the Neumann series does not converge (for example, when the norm of the matrix is 1 - such as in the case where the Laplacian operator is employed) our method can still be applied.

2 Formulation

Consider a general description of the Monte Carlo method. Let X be a Banach space of real-valued functions. Let $f = f(x) \in X$ and $u_k = u(x_k) \in X$ be defined in R^n and $L = L(u)$ be a linear operator defined on X.

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

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

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

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

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

A special case is when the corresponding infinite series converges. Then the sum is an element u from the space X which satisfies the equation

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

The truncation error of (2) is thus

$$u_k - u = L^k(u_0 - u). \tag{4}$$

Let $J(u_k)$ be a linear functional that is to be calculated, and consider the spaces

$$T_{i+1} = \underbrace{R^n \times R^n \times \ldots \times R^n}_{i \text{ times}}, \qquad i = 1, 2, \ldots, k \qquad , \tag{5}$$

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

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

$$E\theta_0 = J(u_0), \;\; E(\theta_1/\theta_0) = J(u_1), \ldots, E(\theta_k/\theta_0) = J(u_k). \tag{6}$$

The computational problem then becomes one of calculating repeated realisations of θ_k and combining them into an appropriate statistical estimator for $J(u_k)$. As an approximate value of the linear functional $J(u_k)$, form

$$J(u_k) \approx \frac{1}{N} \sum_{s=1}^{N} \{\theta_k\}_s \qquad , \tag{7}$$

where $\{\theta_k\}_s$ is the s-th realisation of the random variable θ_k. The probable error r_N of (7) [EM82] is then

$$r_N = c \; s(\theta_k) N^{-\frac{1}{2}} \qquad , \tag{8}$$

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

Clearly it follows from (8) that if we find a method to reduce $s(\theta_k)$ then the number of realizations to achieve a given probable error is reduced.

There are two interesting special cases of the operator L:

(i) L is an ordinary integral transform

$$L(u) = \int_G k(x,y)u(y)dy\,; \tag{9}$$

(ii) L is a matrix and u and f are vectors.

In the first case equation (3) becomes :

$$u(x) = \int_G k(x,y)u(y)dy + f(x)\,. \tag{10}$$

and the random variable whose mathematical expectation coincides with $J(u)$ is given by

$$\theta[h] = \frac{h(\xi_0)}{p(\xi_0)} \sum_{j=0}^{\infty} Q_j f(\xi_j) \qquad , \tag{11}$$

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

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

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

where I is the identity matrix; $L = (l_{ij})$; $u_0 = (u_1^0, \ldots, u_n^0)$ and matrix $I - L$ is supposed to be non-singular.

It is well known that if all eigenvalues of the matrix L lie within the unit circle of the complex plane there exists a vector u such that

$$u = \lim_{k\to\infty} u_k \ , \tag{13}$$

which satisfies the equation

$$u = Lu + f. \tag{14}$$

Now consider the problem of evaluating the inner product

$$J(u) = (h,u) = \sum_{i=1}^{n} h_i u_i \,, \tag{15}$$

where $h \in R^n$ is a given vector .

The following random variable can now be constructed:

$$\theta[h] = \frac{h_{k_0}}{p_0} \sum_{\nu=0}^{\infty} Q_\nu f_{k_\nu} \ , \tag{16}$$

where

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

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

Consider the following system of linear equations:

$$Au = b \ , \tag{17}$$

where $b = (b_1, \ldots, b_m)$ is an m-dimensional vector.

Clearly it is possible to choose a non singular matrix M such that

$$MA = I - L \tag{18}$$

and

$$Mb = f \tag{19}$$

and then (17) becomes

$$MAu = Mb. \tag{20}$$

The last equation is equivalent to (15). If matrices M and A are both non-singular and L has its eigenvalues all inside the unit circle, then (16) becomes a stationary linear iterative process. As a result the convergence of the Monte Carlo method depends on the truncation error of (12).

3 Convergence and mapping

To analyse the convergence of the Monte Carlo method consider the functional equation:

$$u - \lambda Lu = f, \tag{21}$$

where λ is some parameter. Define a resolvent operator R_λ by equation

$$I + \lambda R_\lambda = (I - \lambda L)^{-1}, \tag{22}$$

where I is identity operator.

Let $\lambda_1, \lambda_2, \ldots$ be the characteristic values (or eigenvalues) of equation (21), where

$$|\lambda_1| \leq |\lambda_2| \leq \ldots$$

Monte Carlo algorithms are based on representation

$$u = (I - \lambda L)^{-1} f = f + \lambda R_\lambda f, \tag{23}$$

where

$$R_\lambda = L + \lambda L^2 + \ldots \ , \tag{24}$$

Obviously the resolvent operator R_λ exists if the sequence (24) converges. The systematic error of the presentation (24) when m terms are used is

$$r_s = O[(|\lambda|/|\lambda_1|)^{m+1} m^{\rho-1}], \tag{25}$$

where ρ is the multiplicity of the roots of λ_1.

From (25) is follows that when λ is approximately equal to λ_1 the sequence (24) (and the corresponding Monte Carlo algorithm) converges slowly. When $\lambda \geq \lambda_1$ the algorithm does not converge. Obviously, the representation (24) can be used for $\lambda : |\lambda| < |\lambda_1|$.

Thus, there are two problems to consider.

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

and

Problem 2. How can a Monte Carlo algorithm be constructed, which corresponds to the resolvent of

$$(I - \lambda L)^{-1} f$$

when the sequence (24) does not converge.

To answer these questions we apply a mapping of the spectral parameter λ in (21) . The method follows a similar approach used by E. Gursa, L. Kantorovich & V. Krilov and K. Sabelfeld [Sa89] for integral equations. In Kantorovich & Krilov the mapping approach is used for solving some problems of numerical analysis. To extend these results we have to show that the mapping approach can be extended for any linear operators (including matrices).

Consider the problem of constructing the solution of (21) for $\lambda \in D$ and $\lambda \neq \lambda_k, k = 1, 2, \ldots$, where the domain D lies inside the definition domain of the $R_\lambda f$, such that all characteristic values are outside of the domain D. In the neighbourhood of the point $\lambda = 0$ ($\lambda = 0 \in D$) the resolvent can be expressed by the series

$$R_\lambda f = \sum_{k=0}^{\infty} c_k \lambda^k, \tag{26}$$

where

$$c_k = L^{k+1} f. \tag{27}$$

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

$$\lambda = \psi(\alpha) = a_1 \alpha + a_2 \alpha^2 + \ldots, \tag{28}$$

maps the domain Δ into D. It is possible to use the resolvent:

$$R_{\psi(\alpha)} f = \sum_{j=0}^{\infty} b_j \alpha^j \ , \tag{29}$$

where

$$b_j = \sum_{k=1}^{j} d_k^{(j)} c_k \tag{30}$$

and

$$d_k^{(j)} = \frac{1}{j!}\left[\frac{\partial^j}{\partial\alpha^n}[\psi(\alpha)]^k\right]_{\alpha=0}. \tag{31}$$

Clearly, the domain D can be so chosen that it is possible to map the value $\lambda = \lambda_*$ into point $\alpha = \alpha_* = \psi^{-1}(\lambda_*)$ for which the sequence (29) converges. Hence the solution of the functional equation (21) can be presented in the form:

$$u = f + \lambda_* R_{\psi(\alpha_*)} f, \tag{32}$$

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 (24) converges slowly.

For example, assume that all eigenvalues λ_k are real and $\lambda_k \in (-\infty, -a]$, where $a > 0$. Consider a mapping for the case of interest ($\lambda = \lambda_* = 1$):

$$\lambda = \psi(\alpha) = \frac{4a\alpha}{(1-\alpha)^2}. \tag{33}$$

The sequence $R_{\psi(\alpha)}f$ for the mapping (33) converges absolutely and uniformly.

In Monte Carlo calculations we keep m terms of the sequence (29):

$$R_{\lambda_*} f \approx \sum_{k=1}^{m} b_k \alpha_k^k = \sum_{k=1}^{m} \alpha_*^k \sum_{i=1}^{k} d_i^{(k)} c_i = \sum_{k=1}^{m} g_k^{(m)} c_k, \tag{34}$$

where

$$g_k^{(m)} = \sum_{j=k}^{m} d_k^{(j)} \alpha_*^j. \tag{35}$$

and the coefficients

$$d_k^{(j)} = (4a)^k q_{k,j} \tag{36}$$

and $g_k^{(m)}$ can be calculated at advance. The calculated coefficients $d_k^{(j)}$ for the mapping (33) are presented in table 1 (for $k, j \le 9$) .

It is easy to see that coefficients $q_{k,j}$ are combinations of the following type:

$$q_{k,j} = C_{k+j-1}^{2k-1}. \tag{37}$$

For calculating iterations $c_k = L^{k+1} f$ a Monte Carlo method must be used. It is easy to see that

$$g_k^{(m)} \le 1. \tag{38}$$

In fact

$$\lim_{m\to\infty} g_k^{(m)} = 1 \tag{39}$$

and $d_k^{(n)}$ and α_*^n are both positive. So $g_k^{(m)}$ increases for any fixed k when m increases. Thus, $|g_k^{(m)}| \le 1$ for any $k, m = 1, 2, \ldots$

The mapping (33) creates the following Monte Carlo iteration process

$$u_0 = f$$

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

Table 1: **Table of the coefficients** $q_{k,j} = (4a)^{-k} d_k^{(j)}$ **for** $k, j \leq 9$

$$\begin{aligned} u_1 &= 4aLu_0 \\ u_2 &= 4aLu_1 + 2u_1 \\ u_3 &= 4aLu_2 + 2u_2 - u_1 \\ u_j &= 4aLu_{j-1} + 2u_{j-1} - u_{j-2}, \;\; j > 2. \end{aligned} \tag{40}$$

and from (40) we have

$$u^{(m)} = 4a\alpha L u^{(m-1)} + 2\alpha u^{(m-1)} - \alpha^2 u^{(m-2)} + f(1-\alpha^2), \;\; m > 2. \tag{41}$$

4 The Method

Suppose we have a Markov chain with m states. The random trajectory (chain) T_i of length i starting in the state k_0 is defined as follows:

$$k_0 \to k_1 \to \cdots \to k_j \to \cdots \to k_i, \tag{42}$$

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

Assume that

$$P(k_0 = \alpha) = p_\alpha, \quad P(k_j = \beta | k_{j-1} = \alpha) = p_{\alpha\beta}, \tag{43}$$

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. We require that

$$\sum_{\alpha=1}^{n} p_\alpha = 1, \quad \sum_{\beta=1}^{n} p_{\alpha\beta} = 1, \;\; \text{for any} \;\; \alpha = 1, 2, ..., n. \tag{44}$$

Suppose the distributions created from the density probabilities p_α and $p_{\alpha\beta}$ are acceptable, according to the following definition:

The distribution $(p_{k_1}, ..., p_{k_m})^t$ is acceptable to vector h, and similarly that the distribution $p_{k_{\nu-1}, k_\nu}$ is acceptable to L [So73], if

$$\begin{cases} p_{k_\nu} > 0 \;\; \text{when} \;\; h_{k_\nu} \neq 0 \\ p_{k_\nu} \geq 0 \;\; \text{when} \;\; h_{k_\nu} = 0 \end{cases} \tag{45}$$

$$\begin{cases} p_{k_{\nu-1},k_\nu} > 0 \ \text{ when } \ l_{k_{\nu-1},k_\nu} \neq 0 \\ p_{k_{\nu-1},k_\nu} \geq 0 \ \text{ when } \ l_{k_{\nu-1},k_\nu} = 0 \end{cases} \tag{46}$$

Now consider the problem of evaluating the inner product $J(u) = (h, u) = \sum_{\alpha=1}^{n} h_\alpha u_\alpha$ of a given vector h with the vector solution of the system (15).

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

$$\theta^*[h] = \frac{h_{k_0}}{p_0} \sum_{\nu=0}^{m} g_\nu^{(m)} Q_\nu f_{k_\nu} \qquad , \tag{47}$$

where $Q_0 = 1$; $g_0^{(m)} = 1$;

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

($k_0, k_1, k_2, \ldots$ is a Markov chain in R^n with initial density function p_{k_0} and transition density function $p_{k_{\nu-1},k_\nu}$) and coefficients $g_j^{(m)}$ are defined by (35) for $j \geq 1$.

The following theorem can be proved:

Theorem 4.1 *Consider matrix A, whose Neumann series (24) does not converge. Let (33) be the required mapping, so that the presentation (34) exists. Then*

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

Sketch of proof:

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

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

Now calculate the mathematical expectation of the random variable

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

From the definition of the mathematical expectation and presentations (45), (46), (47) it follows:

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

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

The existence and convergence of the sequence (35) ensures the following presentations:

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

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

This theorem permits use of the random variable $\theta_m^*[h]$ for calculating the inner product (15).

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

$$h=e(r)=(0,...,0,1,0,...,0), \tag{48}$$

where the one is in the "r"th place. It follows that

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

and the corresponding Monte Carlo method is given by

$$u_r\approx\frac{1}{N}\sum_{s=1}^{N}\theta_m^*[e(r)]_s, \tag{49}$$

where N is the number of chains and

$$\theta_m^*[e(r)]_s=\sum_{\nu=0}^{m}g_\nu^{(m)}Q_\nu f_{k_\nu}; \tag{50}$$

$$Q_\nu=\frac{l_{rk_1}l_{k_1k_2}\dots l_{k_{\nu-1}k_\nu}}{p_{rk_1}p_{k_1k_2}\dots p_{k_{\nu-1}p_\nu}}. \tag{51}$$

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

$$L=I-A, \tag{52}$$

where I is the identity matrix. Clearly, the inverse matrix is given by

$$C=\sum_{i=0}^{\infty}L^i, \tag{53}$$

which converges if $\|L\|<1$. If this condition is not satisfied or if the corresponding Neumann series converges slowly, we can use the same technique for accelerating the convergence of the method.

To estimate the element $c_{rr'}$ of the inverse matrix C, let the vector f (21) be the following unit vector

$$f_{r'}=e(r'). \tag{54}$$

Theorem 4.2 *Consider matrix A, whose Neumann series (24) does not converge. Let (33) be the required mapping, so that presentation (34) exists. Then*

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

Sketch of proof:

The proof is similar to the proof of Theorem 4.1, but in this case we need to consider an unit vector $e(r)$ instead of vector h and vector $e(r')$ instead of $f_{k\nu}$:

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

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

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

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

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

that is:

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

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

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

This theorem permits use of the following Monte Carlo method for calculating elements of the inverse matrix C:

$$c_{rr'} \approx \frac{1}{N}\sum_{s=1}^{N}\left[\sum_{(\nu|k_\nu=r')}^{m} g_\nu^{(m)}\frac{l_{rk_1}l_{k_1k_2}\ldots l_{k_{\nu-1}k_\nu}}{p_{rk_1}p_{k_1k_2}\cdots p_{k_{\nu-1}p_\nu}}\right], \tag{55}$$

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

$$W_\nu^{(m)} = g_\nu^{(m)}\frac{l_{rk_1}l_{k_1k_2}\ldots l_{k_{\nu-1}k_\nu}}{p_{rk_1}p_{k_1k_2}\cdots p_{k_{\nu-1}p_\nu}} \tag{56}$$

for which $k_\nu = r'$ are included in the sum (52).

Since $W_\nu^{(m)}$ is included only into the corresponding sum for $r' = 1, 2, \ldots, n$, 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 is used later.

5 Balancing of errors

Two types of errors, systematic and stochastic, can occur in Monte Carlo methods, and achieving a balance between these two types of error is necessary. Clearly, to obtain good results the stochastic (probable) error must be approximately equal to the systematic one and so

$$r_N = O(r_s).$$

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.

Let us consider the case when our method is applied to **Problem 1**. Using the procedure of mapping and a random variable , defined by (52), we accelerate the convergence of the method proposed by Curtiss [C54],[C56]. This means, that for a fixed number of steps m,

$$r_s(m) < r_s^{(C)}(m), \tag{57}$$

where $r_s^{(C)}(m)$ is the systematic error of the Curtiss method and $r_s(m)$ is the systematic error of our method. A similar inequality holds for the probable errors. From $g_k^{(m)}$ in (52) it follows that

$$\sigma(\theta^*) < \sigma(\theta) \tag{58}$$

and thus

$$r_N(\sigma(\theta^*)) < r_N^{(C)}(\sigma\theta) \tag{59}$$

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

Next consider the general error

$$R = r_N(\sigma) + r_s(m) \tag{60}$$

for matrix inversion by our Monte Carlo approach. Let R be fixed. Obviously from (57) and (58) 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 consider the problem of matrix inversion for a fixed general error R, we have:

$$R = R^{(C)} = r_N^{(C)}(\sigma) + r_s^{(C)}(m) = c_N r_N(\sigma) + c_s r_s(m). \tag{61}$$

The last presentation shows that in our method we can reduce one (say N) or both parameters N and $T = E(m)$. In fact,

$$\begin{aligned} c\sigma(\theta)/N_c^{1/2} + r_s^{(C)}(m) &= cc_N\sigma(\theta^*)/N_c^{1/2} + c_s r_s(m) \\ &= c\sigma(\theta^*)/N^{1/2} + r_s(m), \end{aligned} \tag{62}$$

or

$$c\sigma(\theta^*)/N^{1/2} = cc_N\sigma(\theta^*)/N_c^{1/2} + (c_s - 1)r_s(m) \tag{63}$$

and

$$\frac{1}{N^{1/2}} = \frac{c_N}{N_c^{1/2}} + \frac{(c_s - 1)r_s(m)}{c\sigma(\theta^*)}, \tag{64}$$

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

Denote by b the following strongly positive variable:

$$b = \frac{(c_s - 1)r_s(m)}{c\sigma(\theta^*)} > 0. \tag{65}$$

From (64) and (65)

$$N = \frac{N_c}{\left(c_N + bN_c^{1/2}\right)^2}. \tag{66}$$

Alltough result (66) is an exact result, in practice it may be difficult to estimate $r_s(m)$ exactly. In such cases by employing the conditions $b > 0$ (65), we can obtain the following estimate for N :

$$N < \frac{N_c}{c_N^2}. \tag{67}$$

This last result shows that for our method the number of realizations N can be at least c_N^2 times less than the number of realizations N_c of the existing method.

6 Discussion

Consider some estimates of N and T. Using an almost optimal frequency function, and according to the principal of collinearity of norms [ID91] and [MAD93], we choose $p_{\alpha\beta}$ proportional to the $|a_{\alpha\beta}|$. Depending on estimates of the convergence of the Neumann series, one of the following stopping rules can be selected to drop Markov chains:

(i) when $|W_\nu^{(m)}| < \delta$

or

(ii) when a chain enters an absorbing state [C54, C56, FL50].

In the first case this leads to a small bias.

For the **Monte Carlo method without absorbing states (MWA)** ($f_{k_j} = \delta_{k_j\beta}$ in (56), if $\alpha\beta$-th entry of inverse matrix is computed) the bounds on T and $D\Theta^*$ are

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

and

$$D\Theta^* \leq \frac{\|\varphi\|^2}{(1 - \|L\|)^2} \leq \frac{1}{(1 - \|L\|)^2}.$$

Consider **Monte Carlo methods with absorbing states (MA)** [C54, C56, FL50], where $\eta^*[g]$ means a random variable $\eta_T[g]$ (T is the length of the chain when absorption take place) taken over infinitely long Markov chain.

The bounds on T and $D\eta^*[g]$ [C54, C56] 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

$$D\eta^*[g] \leq \frac{1}{(1 - \|L\|)^2}.$$

According to the Central Limit Theorem

$$N \geq \frac{0.6745^2}{\epsilon^2} D\eta^*[g]$$

for the given error ϵ. Thus

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

is a lower bound on N.

ϵ	δ	$\|L\|$	T_{min}	T_{max}	N	$\|L\|$	T_{min}	T_{max}	N
0.4	0.1	0.9	7	16	284 - 2500	0.5	0	4	11 - 100
0.3	0.1	0.9	6	17	505 - 4444	0.5	0	4	20 - 177
0.2	0.1	0.9	6	17	1137 - 10000	0.5	0	4	45 - 399
0.1	0.1	0.9	7	17	4549 - 40000	0.5	0	4	181 - 1599
0.05	0.05	0.9	10	21	18198 - 160000	0.5	0	5	727 - 6398

Table 2: Method without absorbing states

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

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

In both methods T is bounded by $O(\sqrt{N})$, since in MWA

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

and in MA

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

Results in [DT93] show that for sufficiently large N, $T \approx \sqrt{N}$. Results of experiments for different matrix norms $\|L\|$ and precision ϵ are outlined in Table 2.

From the above results and discussion, it is clear that the parameters N and T can be reduced by using our new method. N can be at least less than N_c/c_N^2, i.e. c_N^2 times less than the number of realizations N_c of the existing method. Furthermore, if we can ensure such mapping that $\|L\| \leq const < 1 - \epsilon_1$, where $1/2 \leq \epsilon_1 < 1$ we can bound the parameter T by some constant and therefore reduce dramatically the computations.

7 Conclusion

In this paper we introduce a new mathematical framework, that develops an alternative Monte Carlo method, which produces a reduction in number of chains N and chain length

T making the algorithms applicable to smaller size matrices. In particular, we present a new iterative Monte Carlo method with high convergence rate. In comparison with existing Monte Carlo methods, which are stationary iterative methods of degree $j = 1$, the new method has degree $j = 2$. This method allows use of a smaller number of iterations (and smaller T) for reaching a given error. In addition, when the Neumann series does not converge (for example, when the norm of the matrix is 1 such as when the Laplacian operator is employed) our method can still be applied.

A comparison with iterative Monte Carlo methods with degree one for the same problems is made. It is shown that both the number of chains N, and their length T can be reduced for a given error. For example, N can be estimated by $N = N_c/\left(c_N + bN_c^{1/2}\right)^2$, where N_c is number of chains in the existing degree one method. Since $b > 0$, $N < N_c/c_N^2$ Thus for our method the number of realizations N can be at least c_N^2 times less than the number of realizations N_c of the existing method.

Furthermore if we can ensure that $\|L\| \leq const < 1 - \epsilon_1$, where $1/2 \leq \epsilon_1 < 1$, we can bound the parameter T by some constant and therefore reduce dramatically the number of computations.

The presented method has the following advantages: In respect to parallel implementations, it will lead to competitive efficient Monte Carlo methods for solving the above problems for matrices with smaller sizes; for regular array designs, this will lead to faster arrays than those based on the existing direct methods.

Acknoledgements: The authors are grateful to the Royal Society, UK for the financial support and to Professor G.M. Megson for the fruitful discussions.

References

[AL96] V.Alexandrov and S. Lakka *Comparison of three Parallel Monte Carlo Methods for Matrix Inversion*, Proc. of EUROPAR96, Lyon, France, Vol II, pp. 72-80.

[AM96] V.Alexandrov and G.M. Megson *Solving Sytem of Linear algebraic Equations by Monte Carlo Method on Regular Arrays*, Proc. of PARCELLA96, 16-20 September, Berlin, Germany, pp. 137-146, 1996.

[BT88] Bertsekas D.P. and Tsitsiklis , *Parallel and Distributed Computation* , Prentice Hall, 1989

[C56] Curtiss J.H. *A Theoretical Comparison of the Efficiencies of two classical methods and a Monte Carlo method for Computing one component of the solution of a set of Linear Algebraic Equations.* ,Proc. Symposium on Monte Carlo Methods , John Wiley and Sons, 1956, pp.191-233.

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

[D88] Delosme J.M., *A parallel algorithm for the algebraic path problem* Parallel and Distributed Algorithms, eds. Cosnard et all, Bonas, France, 1988, pp67-78.

[ID91] Dimov I. *Minimization of the Probable Error for Some Monte Carlo methods*. Proc. Int. Conf. on Mathematical Modeling and Scientific Computation, Varna, 1991

[DT90] I. Dimov, O. Tonev, *Performance Analysis of Monte Carlo Algorithms for Some Models of Computer Architectures*, International Youth Workshop on Monte Carlo Methods and Parallel Algorithms - Primorsko (Eds. Bl. Sendov, I. Dimov), **World Scientific**, Singapore, 1990, 91-95.

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

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

[EA89] El-Amawy A., A Systolic Architecture for Fast Dense Matrix Inversion, *IEEE Transactions on Computers*, Vol.C-38, No3, pp.449-455, 1989.

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

[FL50] Forsythe G.E. and Leibler R.A. *Matrix Inversion by a Monte Carlo Method*, MTAC Vol.4 , 1950 , pp. 127-129.

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

[K84] Kung S.Y., *On Supercomputing with systolic wavefront array processors* , IEEE Proc. Vol.72, pp. 867–884, 1984.

[KL78] Kung H.T. and C.E.Leiserson, *Systolic Arrays for VLSI* , in Sparse Matrix Proceeding 1978, SIAM, 1979, pp. 256–282.

[KLL87] Kung S.Y., S.C.Lo and P.S.Lewis, Optimal Systolic Design for the Transitive Closure and shortest path problems, *IEEE Transactions on Computers* ,

[L95] S. Lakka, Parallel Matrix Inversion Using a Monte Carlo Method, MSc. Dissertation, University of Liverpool, September 1995.

[MAD93] G.M. Megson, V.N. Aleksandrov, I.T. Dimov, *Systolic Matrix Inversion Using a Monte Carlo Method*, **Journal of Parallel Algorithms and Applications, Vol 3**, No 3/4, pp. 311-330.

[MAD94a] G.M. Megson, V.N. Aleksandrov, I.T. Dimov, *A Fixed Sized Regular Array for Matrix Inversion by Monte Carlo Method*, **Adv. in Numerical Methods and Applications , World Scientific**, Proc. of the III Conf. on Num.Meth and Appl., 1994, pp. 255–264.

[M92] Megson G.M., *A Fast Faddeev Array*, **IEEE Transactions on Computers**, Vol.41, N012, December 1992, pp.1594-1600.

[MU49] N. Metropolis, S. Ulam, *The Monte Carlo Method*, **J. of Amer. Statist. Assoc., 44**, (1949), pp. 335–341.

[Qu87] M.J. Quinn, *Designing Efficient Algorithms for Parallel Computers*, **McGraw-Hill**, 1987.

[RM93] Rapanotti L. , G.M.Megson, *Pre-processing in SADE*, TR-431, Dept. Computing Sci., Univ. of Newcastle upon Tyne.

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

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

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