Relaxed Monte Carlo Linear Solver

Chih Jeng Kenneth Tan[1] and Vassil Alexandrov[2]

[1] School of Computer Science
The Queen's University of Belfast
Belfast BT7 1NN
Northern Ireland
United Kingdom
cjtan@acm.org
[2] Department of Computer Science
The University of Reading
Reading RG6 6AY
United Kingdom
v.n.alexandrov@rdg.ac.uk

Abstract. The problem of solving systems of linear algebraic equations by parallel Monte Carlo numerical methods is considered. A parallel Monte Carlo method with relaxation is presented. This is a report of a research in progress, showing the effectiveness of this algorithm. Theoretical justification of this algorithm and numerical experiments are presented. The algorithms were implemented on a cluster of workstations using MPI.
Keyword: Monte Carlo method, Linear solver, Systems of linear algebraic equations, Parallel algorithms.

1 Introduction

One of the more common numerical computation task is that of solving large systems of linear algebraic equations

$$Ax = b \tag{1}$$

where $A \in \mathbb{R}^{n \times n}$ and $x, b \in \mathbb{R}^n$. A great multitude of algorithms exist for solving Equation 1. They typically fall under one of the following classes: direct methods, iterative methods, and Monte Carlo methods. Direct methods are particularly favorable for dense A with relatively small n. When A is sparse, iterative methods are preferred when the desired precision is high and n is relatively small. When n is large and the required precision is relatively low, Monte Carlo methods have been proven to be very useful [6,4,15,1].

As a rule, Monte Carlo methods are not competitive with classical numerical methods for solving systems of linear algebraic equations, if the required precision is high [13].

In Monte Carlo methods, statistical estimates for the components of the solution vector x are obtained by performing random sampling of a certain random

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1289–1297, 2001.

variable whose mathematical expectation is the desired solution [14,18]. These techniques are based on that proposed by von Neumann and Ulam, extended by Forsythe and Liebler [13,9].

Classical methods such as non-pivoting Gaussian Elimination or Gauss-Jordan methods require $\mathcal{O}(n^3)$ steps for a $n \times n$ square matrix [2]. In contrast, to compute the full solution vector using Monte Carlo the total number of steps required is $\mathcal{O}(nNT)$, where N is the number of chains and T is the chain length, both quantities independent of n and bounded [1]. Also, if only a few components of x are required, they can be computed without having to compute the full solution vector. This is a clear advantage of Monte Carlo methods, compared to their direct or iterative counterpart.

In addition, even though Monte Carlo methods do not yield better solutions than direct or iterative numerical methods for solving systems of linear algebraic equations as in Equation 1, they are more efficient for large n. Also, Monte Carlo methods have been known for their embarrassingly parallel nature. Parallelizing Monte Carlo methods in a coarse grained manner is very often straightforward. This characteristic of Monte Carlo methods has been noted even in 1949, by Metropolis and Ulam [12].

2 Stochastic Methods for Solving Systems of Linear Algebraic Equations

Consider a matrix $A \in \mathbb{R}^{n \times n}$ and a vector $x \in \mathbb{R}^{n \times 1}$. Further, A can be considered as a linear operator $A\left[\mathbb{R}^{n \times 1} \rightarrow \mathbb{R}^{n \times 1}\right]$, so that the linear transformation

$$Ax \in \mathbb{R}^{n \times 1} \tag{2}$$

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

The linear transformation in Equation 2 is used in iterative Monte Carlo algorithms, and the linear transformation in Equation 2 is also known as the iteration. This algebraic transform plays a fundamental role in iterative Monte Carlo algorithms.

In the problem of solving systems of linear algebraic equations, the linear transformation in Equation 2 defines a new vector $b \in \mathbb{R}^{n \times 1}$:

$$Ax = b, \tag{3}$$

where A and b are known, and the unknown solution vector x is to be solved for. This is a problem often encountered as subproblems on in various applications such as solution of differential equations, least squares solutions, amongst others.

It is known that system of linear algebraic equation given by Equation 3, can be rewritten in the following iterative form [2,18,4]:

$$x = Lx + b, \tag{4}$$

where

$$(I - L) = A. \tag{5}$$

Assuming that $\|L\| < 1$, and $x^0 \equiv 0$, the von Neumann series converges and the equation

$$\lim_{k\to\infty} x^{(k)} = \lim_{k\to\infty} \sum L^m b = (I - L)^{-1} b = A^{-1} b = x \tag{6}$$

holds.

Suppose now $\{s_1, s_2, \ldots, s_n\}$ is a finite discrete Markov chains with n states. At each discrete time $t = 0, 1, \ldots, N$, a chain S of length T is generated:

$$k_0 \to k_1 \to \ldots \to k_j \to \ldots \to k_T$$

with $k_j \in \{s_1, s_2, \ldots, s_n\}$ for $j = 1, \ldots, T$.

Define the probability that the chain starts in state s_α,

$$\mathbf{P}\ [k_0 = s_\alpha] = p_\alpha \tag{7}$$

and the transition probability to state s_β from state s_α

$$\mathbf{P}\ [k_j = s_\beta | k_{j-1} = s_\alpha] = p_{\alpha\beta} \tag{8}$$

for $\alpha = 1, \ldots, n$ and $\beta = 1, \ldots, n$.

The probabilities $p_{\alpha\beta}$ thus define the transition matrix P. The distribution $(p_1, \ldots, p_n)^T$ is said to be acceptable to vector h, and similarly that the distribution $p_{\alpha\beta}$ is acceptable to L, if [14]

$$\begin{cases} p_\alpha > 0 \text{ when } h_\alpha \neq 0 \\ p_\alpha \geq 0 \text{ when } h_\alpha = 0 \end{cases} \text{ and } \begin{cases} p_{\alpha\beta} > 0 \text{ when } l_{\alpha\beta} \neq 0 \\ p_{\alpha\beta} \geq 0 \text{ when } l_{\alpha\beta} = 0 \end{cases} \tag{9}$$

Define the random variables W_j according to the recursion

$$W_j = W_{j-1} \frac{l_{k_{j-1}k_j}}{p_{k_{j-1}k_j}},\ W_0 \equiv 1 \tag{10}$$

The random variables W_j can also be considered as weights on the Markov chain.

Also, define the random variable

$$\eta_T(h) = \frac{h_{k_0}}{p_{k_0}} \sum_{j=0}^{T-1} W_j b_{k_j}. \tag{11}$$

From Equation 6, the limit of $\mathbf{M}\ [\eta_T\,(h)]$, the mathematical expectation of $\eta_T\,(h)$ is

$$\mathbf{M}\ [\eta_T\,(h)] = \left\langle h, \sum_{m=0}^{T-1} L^m b \right\rangle = \left\langle h, x^{(T)} \right\rangle \Rightarrow \lim_{T\to\infty} \mathbf{M}\ [\eta_T\,(h)] = \langle h, x \rangle \tag{12}$$

Knowing this, one can find an unbiased estimator of $\mathbf{M}\ [\eta_\infty\,(h)]$ in the form

$$\theta_N = \frac{1}{N} \sum_{m=0}^{N-1} \eta_\infty\,(h) \tag{13}$$

Consider functions $h \equiv h^j = (0, 0, \ldots, 1, \ldots, 0)$, where $h_i^j = \delta_i^j$ is the Kronecker delta. Then

$$\langle h, x \rangle = \sum_{i=0}^{n-1} h_i^j x_i = x_j. \tag{14}$$

It follows that an approximation to x can be obtained by calculating the average for each component of every Markov chain

$$x_j \approx \frac{1}{N} \sum_{m=0}^{N-1} \theta_T^m \left[h^j\right] . \tag{15}$$

In summary, N independent Markov chains of length T is generated and $\eta_T(h)$ is calculated for each path. Finally, the j-th component of x is estimated as the average of every j-th component of each chain.

3 Minimal Probable Error

Let I be any functional to be estimated by Monte Carlo method, θ be the estimator, and n be the number of trials. The probable error for the usual Monte Carlo method is defined as [14]:

$$\mathbf{P}\left[|I - \theta| \geq r\right] = \frac{1}{2} = \mathbf{P}\left[|I - \theta| \leq r\right]. \tag{16}$$

Equation (16) does not take into consideration any additional a priori information regarding the regularity of the solution.

If the standard deviation $(\mathbf{D}[\theta])^{\frac{1}{2}}$ is bounded, then the Central Limit Theorem holds, and

$$\mathbf{P}\left[|I - \theta| \leq x \left(\frac{\mathbf{D}[\theta]}{n}\right)^{\frac{1}{2}}\right] \approx \Phi(x). \tag{17}$$

Since $\Phi(0.6745) \approx \frac{1}{2}$, it is obvious that the probable error is

$$r \approx 0.6745 \left(\mathbf{D}[\theta]\right)^{\frac{1}{2}} . \tag{18}$$

Therefore, if the number of Markov chains N increases the error bound decreases. Also the error bound decreases if the variance of the random variable θ decreases.

This leads to the definition of almost optimal transition frequency for Monte Carlo methods. The idea is to find a transition matrix P that minimize the second moment of the estimator. This is achieved by choosing the probability proportional to the $|l_{\alpha\beta}|$ [6]. The corresponding almost optimal initial density vector, and similarly the transition density matrix $P = \{p_{\alpha\beta}\}_{\alpha,\beta=1}^{n}$ is then called the almost optimal density matrix.

4 Parameter Estimation

The transition matrix P is chosen with elements $p_{\alpha\beta} = \frac{|l_{\alpha\beta}|}{\sum_{\beta}|l_{\alpha\beta}|}$ for $\alpha, \beta = 1, 2, ..., n$. In practice the length of the Markov chain must be finite, and is terminated when $|W_j b_{k_j}| < \delta$, for some small value δ [14]. Since

$$|W_j b_{k_j}| = \left| \frac{l_{\alpha_0\alpha_1} \cdots l_{\alpha_{j-1}\alpha_j}}{\frac{|l_{\alpha_0\alpha_1}|}{\|L\|} \cdots \frac{|l_{\alpha_{j-1}\alpha_j}|}{\|L\|}} \right| |b_{k_j}| = \|L\|^i \|b\| < \delta\,, \tag{19}$$

it follows that

$$T = j \leq \frac{\log\left(\frac{\delta}{\|b\|}\right)}{\log\|L\|} \tag{20}$$

and

$$\mathbf{D}\left[\eta_T(h)\right] \leq \mathbf{M}\left[\eta_T^2\right] = \frac{\|b\|^2}{\left(1-\|L\|\right)^2} \leq \frac{1}{\left(1-\|L\|\right)^2}. \tag{21}$$

According to the Central Limit Theorem,

$$N \geq \left(\frac{0.6745}{\epsilon}\right)^2 \frac{1}{\left(1-\|L\|\right)^2} \tag{22}$$

is a lower bound on N.

5 Relaxed Monte Carlo Method

Here, the Relaxed Monte Carlo method is defined. Consider a matrix $E \in \mathbb{R}^{n\times n}$. Multiply matrix E to both left and right sides of Equation 3, so that

$$EAx = Eb. \tag{23}$$

It is then possible to define an iteration matrix L_r,

$$L_r = I - EA, \tag{24}$$

similar to the iteration matrix L Equation 5. It then follows that

$$x = L_r x + f, \tag{25}$$

where

$$f = Eb. \tag{26}$$

The corresponding von Neumann series converges and

$$x^{(k+1)} = \left(I + L_r + L_r^2 + \cdots + L_r^k\right) f = \sum_{m=0}^{k} L_r^m f \text{ where } L_r^0 \equiv I, \tag{27}$$

where

$$\lim_{k\to\infty} x^{(k)} = \lim_{k\to\infty} \sum L_r^m f = (I - L_r)^{-1} f = (EA)^{-1} f = x. \tag{28}$$

Define E as a (diagonal) matrix such that

$$e_{ij} = \begin{cases} \frac{\gamma}{a_{ij}}, \gamma \in (0,1], & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}. \tag{29}$$

The parameter γ is chosen such that it minimizes the norm of L in order to accelerate the convergence. This is similar to the relaxed successive approximation iterative method with the relaxation parameter γ [5]. Similar approach was presented and discussed by Dimov et al. [5] for iterative Monte Carlo method for solving inverse matrix problems, where the matrices were diagonally dominant. The parameter γ was then changed dynamically during the computation. In contrast, the parameter γ chosen in this case is based on a priori information, so that matrix norm is reduced, preferably to less than 0.5.

Furthermore, a set of parameters, $\{\gamma_i\}_{i=0}^{n}$ can be used in place of a single γ value, to give a desirable norm in each row of L_r. Such a choice will also result in a matrix L_r which is more balanced, in terms of its row norms, $\|L_{ri}\|$.

Following the arguments of Faddeev and Faddeeva [7,8], this Relaxed Monte Carlo method will converge if

$$\gamma_i < \frac{2}{\|A_i\|}, \tag{30}$$

where $\|A_i\|$ is the row norm of the given matrix A. This approach is equally effective for both diagonally dominant and non-diagonally dominant matrices. This is also corroborated by the numerical experiments conducted.

It is obvious that the Relaxed Monte Carlo method can be used in conjunction with either the almost optimal Monte Carlo method or the Monte Carlo method with chain reduction and optimization [16,17,3]. In any case, since the Relaxed Monte Carlo method can be used to reduce the norm of a matrix to a specified value, it can always be used to the effect of accelerating the convergence of the Monte Carlo method in general.

6 Numerical Experiments

A parallel version of the Relaxed Monte Carlo algorithm was developed using Message Passing Interface (MPI) [11,10]. Version 1.2.0 of the MPICH implementation of the Message Passing Interface was used. As the programs were written in C, the C interface of MPI was the natural choice interface.

Tables 1 and 2, show the results for experiments with the Relaxed Monte Carlo method. The matrices used in these experiments were dense (general) randomly populated matrices, with a specified norm. The stochastic error, ϵ, and the deterministic error parameters were both set to 0.01. The `PLFG` parallel pseudo-random number generator [17] was used as the source of randomness for the experiments conducted.

Table 1. Relaxed Monte Carlo method with `PLFG`, using 10 processors, on a DEC Alpha XP1000 cluster.

Data set	Norm	Solution time (sec.)	RMS error	No. chains
100-A1	0.5	0.139	4.76872e-02	454900
100-A2	0.6	0.122	4.77279e-02	454900
100-A3	0.7	0.124	4.78072e-02	454900
100-A4	0.8	0.127	4.77361e-02	454900
100-A5	0.5	0.137	3.17641e-02	454900
100-A6	0.6	0.124	3.17909e-02	454900
100-A7	0.7	0.124	3.17811e-02	454900
100-A8	0.8	0.119	3.17819e-02	454900
100-B1	0.5	0.123	3.87367e-02	454900
100-B2	0.6	0.126	3.87241e-02	454900
100-B3	0.7	0.134	3.88647e-02	454900
100-B4	0.8	0.125	3.88836e-02	454900
100-B5	0.5	0.121	2.57130e-02	454900
100-B6	0.6	0.119	2.57748e-02	454900
100-B7	0.7	0.120	2.57847e-02	454900
100-B8	0.8	0.126	2.57323e-02	454900

The time to solution given in the tables is the actual computation time, in seconds. The time taken to load the data is not taken into account, since for many computational science problems, the data are created on the nodes [1].

7 Acknowledgment

We would like to thank M. Isabel Casas Villalba from Norkom Technologies, Ireland for the fruitful discussions and the MACI project at the University of Calgary, Canada, for their support, providing part of the computational resources used.

References

[1] Alexandrov, V. N. Efficient Parallel Monte Carlo Methods for Matrix Computations. *Mathematics and Computers in Simulation 47*, 2 – 5 (1998), 113 – 122.

[2] Bertsekas, D. P., and Tsitsiklis, J. N. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[3] Casas Villalba, M. I., and Tan, C. J. K. Efficient Monte Carlo Linear Solver with Chain Reduction and Optimization Using `PLFG`. (To be published.), 2000.

[4] Dimov, I. Monte Carlo Algorithms for Linear Problems. In *Lecture Notes of the 9th. International Summer School on Probability Theory and Mathematical Statistics* (1998), N. M. Yanev, Ed., SCT Publishing, pp. 51 – 71.

Table 2. Relaxed Monte Carlo method with `PLFG`, using 10 processors, on a DEC Alpha XP1000 cluster.

Data set	Norm	Solution time (sec.)	RMS error	No. chains
1000-A1	0.5	7.764	1.91422e-02	2274000
1000-A2	0.6	7.973	1.92253e-02	2274000
1000-A3	0.7	7.996	1.93224e-02	2274000
1000-A4	0.8	7.865	1.91973e-02	2274000
1000-A5	0.5	7.743	1.27150e-02	2274000
1000-A6	0.6	7.691	1.27490e-02	2274000
1000-A7	0.7	7.809	1.27353e-02	2274000
1000-A8	0.8	7.701	1.27458e-02	2274000
1000-B1	0.5	7.591	1.96256e-02	2274000
1000-B2	0.6	7.587	1.97056e-02	2274000
1000-B3	0.7	7.563	1.96414e-02	2274000
1000-B4	0.8	7.602	1.96158e-02	2274000
1000-B5	0.5	7.147	1.29432e-02	2274000
1000-B6	0.6	7.545	1.30017e-02	2274000
1000-B7	0.7	7.541	1.31470e-02	2274000
1000-B8	0.8	7.114	1.28813e-02	2274000

[5] Dimov, I., Dimov, T., and Gurov, T. A new iterative monte carlo approach for inverse matrix problem. *Journal of Computational and Applied Mathematics 4*, 1 (1998), 33 – 52.

[6] Dimov, I. T. Minimization of the Probable Error for some Monte Carlo Methods. In *Mathematical Modelling and Scientific Computations* (1991), I. T. Dimov, A. S. Andreev, S. M. Markov, and S. Ullrich, Eds., Publication House of the Bulgarian Academy of Science, pp. 159 – 170.

[7] Faddeev, D. K., and Faddeeva, V. N. *Computational Methods of Linear Algebra*. Nauka, Moscow, 1960. (In Russian.).

[8] Faddeeva, V. N. *Computational Methods of Linear Algebra*. Nauka, Moscow, 1950. (In Russian.).

[9] Forsythe, S. E., and Liebler, R. A. Matrix Inversion by a Monte Carlo Method. *Mathematical Tables and Other Aids to Computation 4* (1950), 127 – 129.

[10] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard*, 1.1 ed., June 1995.

[11] Message Passing Interface Forum. *MPI-2: Extensions to the Message-Passing Interface*, 2.0 ed., 1997.

[12] Metropolis, N., and Ulam, S. The monte carlo method. *Journal of the American Statistical Association 44*, 247 (1949), 335 – 341.

[13] Rubinstein, R. Y. *Simulation and the Monte Carlo Method*. John Wiley and Sons, 1981.

[14] Sobol', I. M. *Monte Carlo Numerical Methods*. Nauka, Moscow, 1973. (In Russian.).

[15] Tan, C. J. K., and Blais, J. A. R. PLFG: A Highly Scalable Parallel Pseudo-random Number Generator for Monte Carlo Simulations. In *High Performance Computing and Networking, Proceedings of the 8th. International Conference on High Performance Computing and Networking Europe* (2000), M. Bubak, H. Afsarmanesh, R. Williams, and B. Hertzberger, Eds., vol. 1823 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 127 – 135.

[16] Tan, C. J. K., Casas Villalba, M. I., and Alexandrov, V. An Improved Monte Carlo Linear Solver Algorithm. (To be published.), 2001.

[17] Tan, C. J. K., Casas Villalba, M. I., and Alexandrov, V. N. Monte Carlo Method for Solution of Linear Algebraic Equations with Chain Reduction and Optimization Using PLFG. In *Proceedings of the 2000 SGI Users' Conference* (2000), M. Bubak, J. Mościński, and M. Noga, Eds., Academic Computing Center, CYFRONET, AGH, Poland, pp. 400 – 408.

[18] Westlake, J. R. *A Handbook of Numerical Matrix Inversion and Solution of Linear Equations.* John Wiley and Sons, 1968.