Available online at www.sciencedirect.com
ScienceDirect
Mathematics and Computers in Simulation xxx (2009) xxx–xxx
MATHEMATICS AND COMPUTERS IN SIMULATION
www.elsevier.com/locate/matcom

Monte Carlo linear solvers with non-diagonal splitting

A. Srinivasan *

Department of Computer Science, Florida State University, Tallahassee, FL 32312, United States

Received 11 July 2007; received in revised form 25 November 2008; accepted 21 March 2009

Abstract

Monte Carlo (MC) linear solvers can be considered stochastic realizations of deterministic stationary iterative processes. That is, they estimate the result of a stationary iterative technique for solving linear systems. There are typically two sources of errors: (i) those from the underlying deterministic iterative process and (ii) those from the MC process that performs the estimation. Much progress has been made in reducing the stochastic errors of the MC process. However, MC linear solvers suffer from the drawback that, due to efficiency considerations, they are usually stochastic realizations of the Jacobi method (a diagonal splitting), which has poor convergence properties. This has limited the application of MC linear solvers. The main goal of this paper is to show that efficient MC implementations of non-diagonal splittings too are feasible, by constructing efficient implementations for one such splitting. As a secondary objective, we also derive conditions under which this scheme can perform better than MC Jacobi, and demonstrate this experimentally. The significance of this work lies in proposing an approach that can lead to efficient MC implementations of a wider variety of deterministic iterative processes.

Keywords: Monte Carlo; Linear solver

1. Introduction

The use of Monte Carlo (MC) in linear algebra dates back to the work of von Neumann and Ulam (described by Forsythe and Leibler [6] in 1950). However, with the development of modern deterministic numerical techniques, MC started losing its appeal in numerical linear algebra. There has been a recent revival of interest in MC linear algebra, partly because of advances in MC techniques, but more importantly, due to the increasing importance of applications where the use of MC techniques is attractive [14]. For example, the use of MC is promising in applications where approximate solutions are sufficient, such as in preconditioning, graph partitioning, information retrieval, and feature extraction. Furthermore, parallel MC is very latency tolerant, and so should be effective in a Grid-like environment. MC can also yield specific components of the solution. In addition, the convergence rate is independent of the size of the matrix.

A major problem with current MC linear solver techniques, which we summarize in Section 2, is that they are fundamentally based on the Jacobi method (a diagonal splitting). We proposed a different iterative process and evaluated it empirically with dense matrices in [11,12], and presented a sparse implementation in [13]. Here, we present a more detailed discussion, including theoretical analysis, a better sparse implementation, and more exhaustive empirical

* Corresponding author. Tel.: +1 850 644 0559; fax: +1 850 644 0058.
E-mail address: asriniva@cs.fsu.edu.

0378-4754/$36.00 © 2009 IMACS. Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.matcom.2009.03.010

testing. We explain the splitting and the dense implementation in Section 3, and discuss the sparse implementation in further detail in Section 4. We then present experimental results in Section 5, and summarize our conclusions in Section 6.

2. Current MC techniques

2.1. Matrix vector multiplication

The basis for MC linear algebra techniques is the ability to perform stochastic matrix-vector multiplication. So we first outline this. Further details can be found in [11].

Consider the matrix $C \in \Re^{n\times n}$ and vector $h \in \Re^n$. We construct "transition-probability" and "weight" matrices P and W satisfying the following constraint

$$C_{ij} = P_{ij} \times W_{ij}, 1 \le i, j \le n, \quad \text{with} \quad P_{ij} \ge 0 \quad \text{and} \quad \sum_{i=1}^{n} P_{ij} = 1, 1 \le j \le n. \tag{1}$$

We similarly define "initial-probability" and "initial-weight" vectors p and w satisfying the following constraint

$$h_i = p_i \times w_i, 1 \le i \le n, \quad \text{with} \quad p_i \ge 0 \quad \text{and} \quad \sum_{i=1}^{n} p_i = 1. \tag{2}$$

MC techniques estimate $C^j h$, $j \ge 0$, where C^j is the jth power of C, by constructing a Markov chain of length j, with initial probabilities given by the vector p, and transition probabilities by P^T. The random walk visits a set of states in $\{1, \ldots, n\}$, and we denote the state visited in the ith step by k_i, $i \in \{1, \ldots, j\}$. The probability of the initial state being α is given by $Prob(k_0 = \alpha) = p_\alpha$ and the transition probability by $Prob(k_i = \alpha | k_{i-1} = \beta) = P_{\alpha\beta}$.

Consider random variables X_i defined as follows: $X_0 = w_{k_0}$, $X_i = X_{i-1} \times W_{k_i k_{i-1}}$. If we let δ denote the Kronecker delta function ($\delta_{ij} = 1$ if $i = j$, and 0 otherwise), then it can be shown [3,5] that $E(X_j \delta_{ik_j}) = (C^j h)_i$, $1 \le i \le n$. Therefore, for each random walk, $X_j \delta_{ik_j}$ can be used to estimate the ith component of $C^j h$. That is, in any random walk, the k_j th component is estimated as X_j, and all the other components are estimated as 0. We perform many random walks, maintaining a running sum for estimates of each component, and finally average over the number of walks. Note that in any single walk, the running sum for only one component needs to be updated.

We will later find it useful to estimate $\sum_{j=0}^{m} C^j h$ through $E(\sum_{j=0}^{m} X_j \delta_{ik_j}) = (\sum_{j=0}^{m} C^j h)_i$. Each random walk gives an estimate for the entire sum, with each step of the random walk updating a particular component of the running sum.

2.1.1. Examples of transition probability choice

Popular choices to satisfy Eqs. (1) and (2) are to either (i) make all probabilities in a given column proportional to the magnitude of the corresponding element, or (ii) make all probabilities in each column equal. For example, if we make probabilities proportional to the magnitude of the elements, then the initial probabilities are given by $p_i = |h_i| / \sum_{j=1}^{n} |h_j|$, and the transition probabilities by $P_{ij} = |C_{ij}| / \sum_{k=1}^{n} |C_{kj}|$. The corresponding weights are given by $w_i = sign(h_i) \times \sum_{i=1}^{n} |h_i|$ and $W_{ij} = sign(C_{ij}) \times \sum_{k=1}^{n} |C_{kj}|$.

2.1.2. Multiple matrices

We can easily extend the above technique to multiplying by more than one matrix. For example, we will find it useful to estimate $\sum_{j=0}^{m} (BC)^j Bh$, where $B, C \in \Re^{n\times n}$ and $h \in \Re^n$. Transition probabilities and weights for C and h are chosen with the constraints given earlier by (1) and (2). Transition probabilities $\hat{P}$ and weights $\hat{W}$ are similarly chosen for B too, and a Markov chain of length $2m + 1$ is used to estimate $\sum_{j=0}^{m} (BC)^j Bh$, in a straight-forward generalization of the above technique. This is described in further detail in [11,12].

2.2. Linear solvers

In order to solve $Ax = b$, $A \in \Re^{n\times n}$ and $x, b \in \Re^n$, the starting point of MC techniques is to split A as $A = N - M$, and write the fixed-point iteration [8] $x^{(m+1)} = N^{-1} M x^{(m)} + N^{-1} b = C x^{(m)} + h$, where $C = N^{-1} M$ and $h = N^{-1} b$.

Please cite this article in press as: A. Srinivasan, Monte Carlo linear solvers with non-diagonal splitting, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.03.010

Then we get $x^{(m)} = C^m x^{(0)} + \sum_{i=0}^{m-1} C^i h$. The initial vector $x^{(0)}$ is often taken to be h for convenience, yielding the Neumann series given below, which converges to the solution as $m \to \infty$ if $\|C\| < 1$.

$$x^{(m)} = \sum_{i=0}^{m} C^i h. \tag{3}$$

MC techniques construct a Markov chain to estimate the sum in (3), with the initial probabilities determined by h, and transition probabilities by C, as explained in Section 2.1.[1]

For effectiveness of the MC technique, efficient determination of C is considered important. Therefore, current MC techniques choose N to be a diagonal matrix, thereby yielding $C = N^{-1}M$ efficiently.[2] This yields, for example, the Jacobi method when N is taken to be the diagonal of A. This perceived need for choosing N to be diagonal has resulted in the iterative schemes underlying current MC techniques having poor convergence properties. Though variance reduction, residual correction, and other techniques have been applied on top of this to get better accuracy, the fundamental limitation is that the MC techniques estimate a quantity that itself does not converge fast.[3]

On the other hand, we note that MC techniques do not have to be based on the best possible iterative technique. Estimates can be obtained fast, and this fact may compensate for the underlying iterative scheme having poor convergence properties. Despite this fact, MC techniques have generally not been competitive with deterministic techniques, except for a limited number of applications. Furthermore, the convergence properties of the underlying iterative scheme restrict the systems to which the current MC techniques can be applied. Therefore, there is a need for MC techniques based on better underlying iterations.

3. Non-diagonal splitting

Diagonal splittings have the following advantages: (i) Determining C is fast, and (ii) C is sparse when A is sparse. In this section, we consider a non-diagonal splitting, and show how its deficiencies with respect to the above two properties can be overcome. We then prove the convergence of this method for a certain class of matrices.

3.1. The splitting

We choose N to be the diagonal and first subdiagonal of A, which we refer to as the *SDI* scheme. (We assume $d_i \neq 0$, $1 \leq i \leq n$, as with the Jacobi method, to ensure that N^{-1} exists.) This yields N of the form

$$N = \begin{pmatrix} d_1 & & & & \\ s_2 & d_2 & & & \\ & s_3 & d_3 & & \\ & & \ddots & & \\ & & & s_n & d_n \end{pmatrix}. \tag{4}$$

N^{-1} is a lower triangular matrix, and just computing $C = N^{-1}M$ would make this splitting non-competitive. So we do not explicitly compute C, but rather, estimate the result of the recurrence $x^{(m+1)} = N^{-1}Mx^{(m)} + N^{-1}b$, which yields

[1] We wish to note that there are many different estimators available [1,2,6,7,15](with the one we mentioned being a popular one, and similar to that introduced by Wasow [15]). Similarly, one may use absorbing or non-absorbing walks (we use the latter), and one may use the direct or adjoint method (we use the latter). However, the fundamental idea behind these alternatives is similar.

[2] For a general N, computing each of N^{-1} and C would be prohibitively expensive.

[3] A modified approach in [4] scales the elements of the matrix, so that Monte Carlo can be used for non-diagonally dominant matrices, provided that the Neumann series, with the scaled matrix, converges.

Please cite this article in press as: A. Srinivasan, Monte Carlo linear solvers with non-diagonal splitting, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.03.010

the following when we take $x^{(0)} = b$.

$$x^{(m)} = \sum_{j=0}^{m} (N^{-1}M)^j N^{-1} b. \tag{5}$$

$x^{(m)}$ can be estimated as shown in Section 2.1.2. The number of steps in each random walk will be twice the number in the current techniques. However, if the method converges faster, then we may compensate for this.

3.2. Dense matrix implementation

The matrix N^{-1} is lower triangular, and in general, the lower triangle is dense. While general matrix inversion takes $O(n^3)$ time, which is prohibitive, we can compute N^{-1} in just $O(n^2)$ time and space as shown below. It is easy to verify [11,12] that

$$N_{ij}^{-1} = \begin{cases} 0 & \text{if } i < j \\ \dfrac{1}{d_i} & \text{if } i = j \\ \dfrac{(-1)^{i-j}}{d_j} \Pi_{k=j+1}^{i} \dfrac{s_k}{d_k} & \text{otherwise} \end{cases} \tag{6}$$

We note the following two points:

- Any element of N^{-1} can be computed in constant time, using $O(n)$ storage and $O(n)$ pre-computation time as follows. Compute and store $T(i)$ [4], $1 \le i \le n$, defined as follows

$$T(i) = \begin{cases} 1 & i = 1 \\ T(i-1)\dfrac{s_i}{d_i} & \text{otherwise} \end{cases} \tag{7}$$

  This takes $O(n)$ space and time. Any element of N^{-1} can be computed in constant time as follows

$$N_{ij}^{-1} = \begin{cases} 0 & \text{if } i < j \\ \dfrac{1}{d_i} & \text{if } i = j \\ \dfrac{(-1)^{i-j}}{d_j} \dfrac{T(i)}{T(j)} & \text{otherwise} \end{cases} \tag{8}$$

- The entire matrix N^{-1} can be computed in $O(n^2)$ time, for example, using Eqs. (7) and (8). We perform such a computation, and make the probability choice (i) from Section 2.1.1.

3.3. Convergence

The Neumann series for the Jacobi method converges for diagonally dominant matrices. We can show that it converges for the SDI method too. In fact, the bounds on its convergence rate, using Gershgorin's theorem, are at least as good as that of Jacobi.

Theorem 1. *If A is row-wise diagonally dominant, then x^m in* Eq. (5) *converges to the true solution as $m \to \infty$.*

Proof. In Appendix B. □

[4] If some $s_i = 0$, then we need to make some modifications, as shown in Appendix A.

Please cite this article in press as: A. Srinivasan, Monte Carlo linear solvers with non-diagonal splitting, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.03.010

If we use absorbing random walks, then we need to show convergence of the stochastic error as the walk length approaches infinity. Though we use non-absorbing walks, it is still useful to shown conditions for such convergence. The condition given below indicates that it converges, provided the subdiagonal entries are not too large. The condition given below appears too restrictive, and empirical tests suggest that the stochastic error converges for much larger subdiagonal entries too, as shown in Section 5.

Theorem 2. *If A is scaled to have diagonal elements of magnitude* 1, *and the scaled matrix is column-wise and row-wise diagonally dominant with* $\max_j \sum_r |m_{rj}| + \max_k |a_{k,k-1}| < 1$, *then the variance of the estimate for each component converges as* $m \to \infty$, *when we use probability choice (i) of Section 2.1.1 for each of* N^{-1} *and* M.

Proof. In Appendix B. □

4. Sparse matrix implementation

We now consider the case where A is sparse. The solution is still estimated using the procedure outlined in Section 3, in particular, using Eq. (5). Since A is sparse, so is M, and the space required for the weight matrix and the transition probability (implicitly stored as data for the alias method [9], which we use for sampling) is proportional to the number of non-zero elements in M. Weight and probability computation for the b vector is as for the dense case, since b is, in general, dense. However, N^{-1} is dense, and we can neither afford $O(n^2)$ computation time to determine it, nor the $O(n^2)$ space to store it.

We first choose a suitable transition probability matrix for N^{-1}, such that it can be sampled fast, and can be represented using $O(n)$ data. On a transition from state j to state i, we can then determine weight $\hat{W}_{ij}$ in constant time using $N^{-1}_{ij} = \hat{P}_{ij}\hat{W}_{ij}$, provided that N^{-1}_{ij} is known. The latter can be computed in constant time, as shown in Eqs. (7) and (8), using $O(n)$ precomputation time and storage. So we just need a suitable transition probability. We show two schemes for that below, assuming that the sub-diagonal entries of A are non-zero. The generalization to zero elements follows from the discussion in Appendix A.

4.1. Equal probabilities

We can assign transition probabilities $\hat{P}$ such that the non-zero elements in each column of N^{-1} have equal probability of occurring. Thus $\hat{P}$ is defined by

$$\hat{P}_{ij} = \begin{cases} 0 & \text{if } i < j \\ \dfrac{1}{n-j+1} & \text{otherwise} \end{cases} \tag{9}$$

We can obtain samples easily in a small constant time by suitably sampling from the uniform distribution. Simulations are performed as with the dense matrix, except that the probabilities for N^{-1} are chosen according to (9), using the corresponding weights.

4.2. Geometric probabilities

In order for the probabilities to be closer to the dense version, we sample from an approximation to the geometric distribution with parameter r, as follows. Define

$$\hat{P}_{ij} = \begin{cases} 0 & \text{if } i < j \\ \approx \text{proportional to} (1-r)^{i-j} & \text{otherwise} \end{cases} \tag{10}$$

A sample from the geometric distribution can be obtained in constant time by first sampling x from the uniform distribution in (0, 1), and then taking $\lceil \log(1-x)/\log(1-r) \rceil$. We choose $r = 1 - \max_k |s_k|$. The actual distribution may be an approximation for the following reason. We need only a finite number of states, and so need some procedure to handle the states obtained from the above process. We can handle this in a variety of ways, for example, omitting states outside the range (at the cost of wasting some computational effort, with the fraction of effort wasted being at

Please cite this article in press as: A. Srinivasan, Monte Carlo linear solvers with non-diagonal splitting, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.03.010

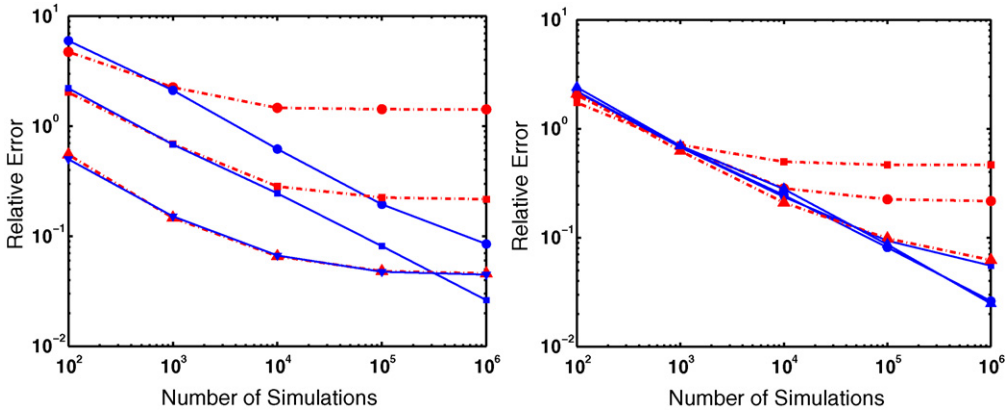

Fig. 1. Plot of relative error vs. number of simulations. The dashed lines are MC Jacobi, and the solid lines are SDI (probability proportional to magnitude of element). (Left) Walk length is 5 in all cases. Circles, SA ($\alpha = 0.3$), squares, A ($\alpha = 0.3$), triangles, G2 ($\alpha = 0.6$). (Right) A ($\alpha = 0.3$) in all cases. Squares, Walk length $m = 2$, circles, $m = 5$, triangles, $m = 10$.

most r) or performing a modulo operation as follows. If k is the state obtained from the geometric distribution, then we can set the actual state to $(k-1) \mod (n-j+1)+1$ when sampling in column j. The experiments reported here are equivalent to taking the former alternative. We have also performed and reported the latter (an approximation) in earlier work.

5. Experimental results

We first wish to compare the effectiveness of MC linear solvers based on the new technique and the conventional Jacobi technique for different numbers of simulations, walk lengths, and extents of diagonal dominance of the matrix A. We also wish to compare the stochastic errors from different probability choices for the new technique. We performed studies with the families of matrices given below. We assume that all the matrices have been scaled to make the diagonal elements 1. The name of each matrix family is enclosed in parentheses below, and followed by a short description.

(A), All elements of the first subdiagonal are set to a parameter α, which controls the extent of diagonal dominance. All other elements are set to $0.5/n$, where $n = 50$ is the number of rows in the matrix. (SA), It is similar to (A), but the superdiagonal elements too are set to α, making the matrix symmetric, and $n = 50$. (RA), It is similar to (A), but the subdiagonal elements are randomly and uniformly distributed in the interval $(\alpha - r, \alpha + r)$, where r is an additional parameter that controls the extent of variation of the subdiagonal elements, and $n = 50$. (G2), Here, a_{ij} is first set to a random value in (0, 1), and then $a_{ij} \leftarrow a_{ij}/\sum_i a_{ij} \times \alpha$, and $n = 100$. In this matrix, the subdiagonal entries are not specially large.

The vector solved for is always set to one with all components equal to 1. The plots compare the relative errors of the different methods, where the relative error is defined as $\|\text{Exact} - \text{Computedsolution}\|_2/\|\text{Exactsolution}\|_2$. When computing the stochastic error, the exact solution is taken to be the vector that results from the deterministic iterative process with the number of iterations set to the walk length. That is, this error determines how close the MC estimate is to the vector it is trying to estimate. When we mention an error in general, the exact solution is taken to be the true solution to the linear system. We computed errors and stochastic errors for each of the above matrices with various walk lengths[5] and number of simulations $N_{sim} \in \{10^2, 10^3, 10^4, 10^5, 10^6\}$. The parameters for matrices were chosen as follows: (A) and (SA) $\alpha \in \{0.1, 0.2, 0.3, 0.4\}$, (RA) $\alpha = 0.2$, $r \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$, and (G2) $\alpha \in \{0.5, 0.6, 0.7, 0.8\}$. We show some samples of these results, which are typical of the trends exhibited.

From Fig. 1 (left), we can see that SDI is better than MC Jacobi for (A) and (SA), and that both are equally good for (G2). The reason for this is that the subdiagonal entries play a significant role in the former two matrices. In the latter, the subdiagonal entries are not large, and so both splitting are essentially identical. We also see the the relative

[5] When we refer to *walk length* in the following discussion, we are referring to the number of terms m in the Neumann series, excluding the first one.

Please cite this article in press as: A. Srinivasan, Monte Carlo linear solvers with non-diagonal splitting, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.03.010

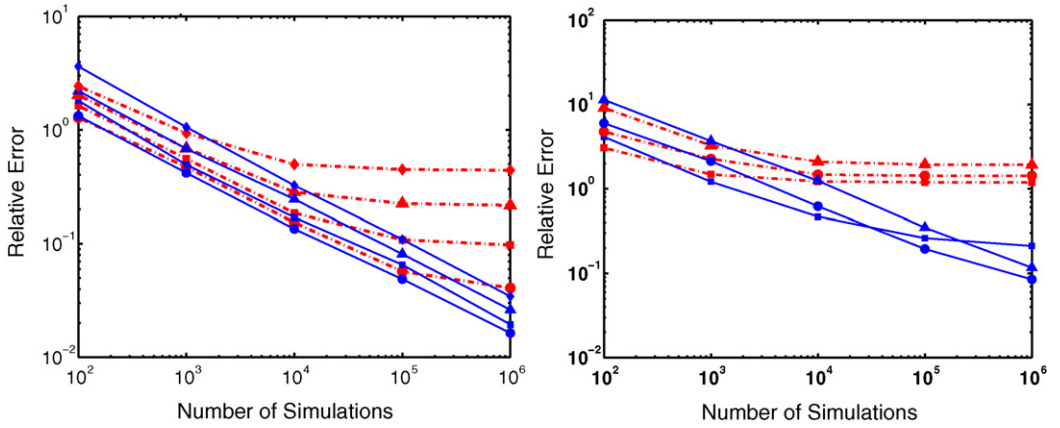

Fig. 2. Plot of relative error vs. number of simulations. The dashed lines are MC Jacobi, and the solid lines are SDI (probability proportional to magnitude of element). (Left) Matrix (A), walk length is 5 in all cases. Circles, $\alpha = 0.1$, squares, $\alpha = 0.2$, triangles, $\alpha = 0.3$, and diamond, $\alpha = 0.4$. (Right) SA ($\alpha = 0.3$) in all cases. Squares, Walk length $m = 2$, circles, $m = 5$, triangles, $m = 10$.

errors are largest for (SA) and lowest for (GA). This is due to the smallest degree of diagonal dominance in (SA), and the largest in (GA).

Fig. 1 (right) compares the two methods with varying walk lengths. All the curves for SDI are almost identical (and visually indistinguishable), suggesting that the error from the iterative process is small, and the error is mainly stochastic. For MC Jacobi, the error decreases as the walk length increases, as expected, until it reaches the same point as SDI, with walk length 10. At this point, the error from the iterative process is small, and since that is the main difference between the two methods, they yield similar results.

Fig. 2 (left) compares the two schemes with different extents of diagonal dominance. As expected, when the extent of diagonal dominance increases, both schemes perform better. SDI performs better than MC Jacobi in all cases. However, the difference is larger when the diagonal dominance is smaller. The reason for this is that the iterative errors are larger with a smaller diagonal dominance, and since SDI improves the iterative error, it performs much better then.

Fig. 2 (right) shows a situation where the Neumann series for Jacobi does not converge, since A is not diagonally dominant. However, the SDI solution converges, suggesting that the convergence conditions in Theorems 1 and 2 can be made less restrictive.

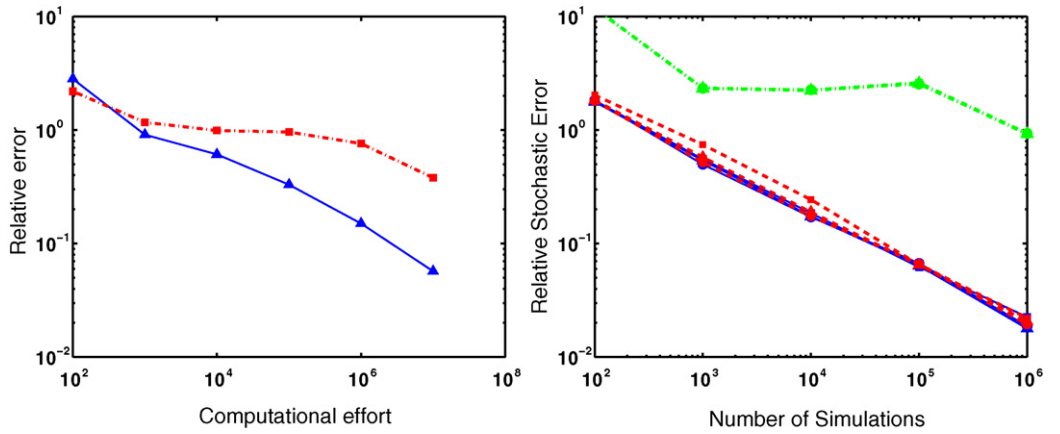

Fig. 3. (Left) Plot of error vs. optimal computational effort for that error. The dashed line is MC Jacobi, and the solid lines is SDI (probability proportional to magnitude of element). Matrix (A) $\alpha = 0.25$. (Right) Comparison of stochastic error for different probability choices for N^{-1} with SDI. RA ($\alpha = 0.2$) and walk length 5 in all cases. Dash-dotted lines (toward the top)—equal probability, dashed lines, geometric distribution, and solid lines, probability proportional to magnitude of element. Circles, $r = 0.1$, triangles, $r = 0.2$, squares, $r = 0.3$.

Please cite this article in press as: A. Srinivasan, Monte Carlo linear solvers with non-diagonal splitting, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.03.010

Fig. 3 (left) determines the optimal computational effort ($(m+1) * N_{sim}$ for MC Jacobi and $2(m+1) * N_{sim}$ for SDI) required to obtain a specified error for each method, and then compares the two. The computational effort accounts for the fact that SDI requires twice the number of steps that MC Jacobi does, to estimate the same number of terms of the Neumann series. We can see that SDI performs much better than Jacobi, except at very small computational efforts. At such small efforts, the stochastic error dominates. Consequently, when variance reduction techniques are used, SDI becomes better at an even earlier point [12].

In Fig. 3 (right), we compare the stochastic errors from three different probability choices. While we expect to use geometric or equal probabilities only with sparse matrices, we compared them with the probability choice used in the dense implementation, on a dense matrix, in order to see if they perform similar to the dense one. In all cases, this error appears to depend primarily on the probability choice, rather than on the extent to which subdiagonal entries vary. The geometric and dense probability perform equally well, while equal probabilities performs relatively poorly.

6. Conclusions

We have proposed efficient MC implementations, dense and sparse, for a non-diagonal splitting, even though it superficially suffers from the disadvantages that have prevented the use of non-diagonal splittings in MC linear solvers. This suggests an approach for MC implementations of a wider variety of stationary iterative techniques. We have also shown, theoretically, that the Neumann series of the new splitting has better bounds on its convergence rate than does Jacobi. We have also empirically demonstrated its effectiveness, in comparison with the conventional Jacobi-based MC linear solver.

Appendix A. Modifications when some $s_i = 0$

As mentioned earlier, we need to make some modifications in case s_i can be zero, $2 \le i \le n$. If $s_k = 0$, then N_{ij}^{-1} will be zero if $i \ge k$ and $j < k$, as can be seen from Eq. (6). In the sparse implementation, we wish to assign probabilities and weights of 0 to such elements. The definition of T in Eq. (7), however, will cause a division of the form 0/0 in Eq. (8) for certain elements. Therefore we need to modify these definitions. The definition of $\hat{P}$ in Eq. (9) too needs to be modified to assign elements with $N_{ij}^{-1} = 0$ a probability of 0, and to increase the probability of non-zero elements suitably.

For each j, $1 \le j \le n$, if there exists a k, $j+1 \le k \le n$, such $s_k = 0$, then define $L(j)$ to be the smallest such k. Otherwise, define $L(j)$ to be $n+1$. We note that the non-zero elements of column j are those in rows j to $L(j)-1$. $L(j)$, $1 \le j \le n$, can be computed in $O(n)$ time and space using the following recurrence

$$L(j) = \begin{cases} n+1 & j = n \\ L(j+1) & s_{j+1} \neq 0 \text{ and } j \neq n \\ j+1 & s_{j+1} = 0 \text{ and } j < n \end{cases} \tag{A.1}$$

Here we start by computing $L(n)$, and proceed in descending order, until we compute $L(1)$.

We are now in a position to modify the computational definition of N^{-1} given by Eq. (8), to set the appropriate elements to 0. However, we also need to modify the definition of T, to avoid division by 0. Note that if we define $T(l) = 1$ when $s_l = 0$, then, the rest of the definition of T in Eq. (7) would be acceptable for the non-zero elements in Eq. (8), since that would involve computing $\alpha_{l+r} \cdots \alpha_{l+s}$ as $(1 \cdot \alpha_{l+1} \cdots \alpha_{l+s})/(1 \cdot \alpha_{l+1} \cdots \alpha_{l+r-1})$ for non-zero elements in the appropriate range. We formalize these with the following definitions

$$T(i) = \begin{cases} 1 & i = 1 \text{ or } s_i = 0 \\ T(i-1)\dfrac{s_i}{d_i} & \text{otherwise} \end{cases} \tag{A.2}$$

Please cite this article in press as: A. Srinivasan, Monte Carlo linear solvers with non-diagonal splitting, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.03.010

Furthermore,

$$N_{ij}^{-1} = \begin{cases} 0 & \text{if } i < j \text{ or } i \geq L(j) \\ \dfrac{1}{d_i} & \text{if } i = j \\ \dfrac{(-1)^{i-j}}{d_j}\dfrac{T(i)}{T(j)} & \text{otherwise} \end{cases} \tag{A.3}$$

The definitions of $\hat{P}$ and $\hat{W}$ are also suitably changed, as follows

$$\hat{P}_{ij} = \begin{cases} 0 & \text{if } i < j \text{ or } i \geq L(j) \\ \dfrac{1}{L(j) - j} & \text{otherwise} \end{cases} \tag{A.4}$$

Also

$$\hat{W}_{ij} = \begin{cases} 0 & \text{if } i < j \text{ or } i \geq L(j) \\ N_{ij}^{-1}(L(j) - j) & \text{otherwise} \end{cases} \tag{A.5}$$

These modifications do not change either the time or space complexities of this technique.

Appendix B. Convergence proofs

Theorem 3. *If A is row-wise diagonally dominant, then x^m in* Eq. (5) *converges to the true solution as $m \to \infty$.*

Proof. We can pre-multiply A and b by D^{-1}, where D is a diagonal matrix consisting of the diagonal entries of A, to get a linear system with diagonal entries all 1, which too is row-wise diagonally dominant. We therefore assume, without loss of generality, that A is a row-wise diagonally dominant matrix with all diagonal entries equal to 1. □

In order to prove the result, it is sufficient to show that the iteration matrix $C = N^{-1}M$ has norm less than 1. We prove this by using Gershgorin's theorem to bound $\|C\|_2$. That is, we show that $\sum_j |c_{ij}| < 1,\ 1 \leq i \leq n$. Note that $\sum_j |c_{ij}| = \sum_j |\sum_k \hat{n}_{ik} m_{kj}| \leq \sum_j \sum_k |\hat{n}_{ik}||m_{kj}|$, where $\hat{n}_{kj} = (N^{-1})_{kj}$. From Eq. (8), we get

$$\sum_k |\hat{n}_{ik}||m_{kj}| = |m_{ij}| + \sum_{k<i} |m_{kj}| \prod_{l=k+1}^{i} |a_{l,l-1}|.$$

Let us define $|a_{10}| = 0$, for notational convenience. From the above equation, we get the following

$$\begin{aligned} \sum_j \sum_k |\hat{n}_{ik}||m_{kj}| &= \sum_j \sum_{k<i} |m_{kj}| \prod_{l=k+1}^{i} |a_{l,l-1}| + \sum_j |m_{ij}| = \sum_{k<i} \left(\sum_j |m_{kj}| \right) \prod_{l=k+1}^{i} |a_{l,l-1}| + \sum_j |m_{ij}| \\ &= \sum_{k<i} (\gamma_k - |a_{k,k-1}|) \prod_{l=k+1}^{i} |a_{l,l-1}| + \gamma_i - |a_{i,i-1}|, \end{aligned}$$

where we define $\gamma_k = \sum_{j \neq k} |a_{kj}| < 1$ (due to row-wise diagonal dominance). Using the fact that the sum above is non-negative, it is straight-forward to prove the following by induction, as shown below.

$$\sum_{k<i} (\gamma_k - |a_{k,k-1}|) \prod_{l=k+1}^{i} |a_{l,l-1}| + \gamma_i - |a_{i,i-1}| \leq \gamma_i. \tag{B.1}$$

Please cite this article in press as: A. Srinivasan, Monte Carlo linear solvers with non-diagonal splitting, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.03.010

The base case is satisfied with equality for $i = 1$. We show that it is true for $i + 1$, assuming that it is true for i.

$$\sum_{k<i+1}(\gamma_k - |a_{k,k-1}|)\prod_{l=k+1}^{i+1}|a_{l,l-1}| + \gamma_{i+1} - |a_{i+1,i}| =$$

$$\sum_{k<i}(\gamma_k - |a_{k,k-1}|)\prod_{l=k+1}^{i+1}|a_{l,l-1}| + (\gamma_i - |a_{i,i-1}|)|a_{i+1,i}| + \gamma_{i+1} - |a_{i+1,i}| =$$

$$\sum_{k<i}(\gamma_k - |a_{k,k-1}|)\prod_{l=k+1}^{i+1}|a_{l,l-1}| - |a_{i,i-1}||a_{i+1,i}| + \gamma_{i+1} - (1-\gamma_i)|a_{i+1,i}| =$$

$$|a_{i+1,i}|\sum_{k<i}(\gamma_k - |a_{k,k-1}|)\prod_{l=k+1}^{i}|a_{l,l-1}| - |a_{i,i-1}||a_{i+1,i}| + \gamma_{i+1} - (1-\gamma_i)|a_{i+1,i}| \leq$$

$\gamma_{i+1} - (1-\gamma_i)|a_{i+1,i}|$ (using the induction hypothesis) $\leq$

γ_{i+1} (proving the induction hypothesis).

Since $\gamma_i < 1$, this proves the theorem. Note that the bound on the norm of the iteration matrix of the Jacobi iteration, using Gershgorin's theorem, is $\max_i \gamma_i$. Thus the *bound* for the SDI scheme is at least as good as that for Jacobi (which does not necessarily imply that the norm too is at least as small).

Theorem 4. *If A is scaled to have diagonal elements of magnitude* 1, *and the scaled matrix is column-wise and row-wise diagonally dominant with* $\max_j \sum_i |m_{ij}| + \max_k |a_{k,k-1}| < 1$, *then the variance of the estimate for each component converges as $m \to \infty$, when we use probability choice (i) of Section 2.1.1 for each of N^{-1} and M.*

Proof. Every two steps of the random walk updates one component of the solution, with one step corresponding to multiplication by M, and the next step corresponding to multiplication by N^{-1}. Let $\tilde{P}_{ij} = \sum_k \hat{P}_{ik} P_{kj}$ denote the probability of transition from state j to state i over the two steps, where $\hat{P}^T$ is the transition probability matrix for N^{-1} and P^T is the transition probability matrix for M. Then, a sufficient condition for convergence of the variance is for the norm of the matrix with elements $c_{ij}^2/\tilde{P}_{ij}$ to be less than 1 [10], where we define an entry with $c_{ij} = \tilde{P}_{ij} = 0$ as 0. In the following discussions, we will assume that we omit such elements, to make the notation clear. □

We prove the convergence by bounding the norm using Gershgorin's theorem. Note that in Theorem B.1, we showed that Gershgorin's theorem bounds the norm of C at less than one under less restrictive conditions than this theorem. Therefore, if we show that $|c_{ij}|/\tilde{P}_{ij} < 1$, then this will imply that the Gershgorin bounds are less than 1 for $c_{ij}^2/\tilde{P}_{ij}$ too.

Let $T = \max_j \sum_i |m_{ij}|$ and $s = \max_k |a_{k,k-1}|$. If we define $N = \max_j \sum_i |\hat{n}_{ij}|$, then $U \leq 1/(1-s)$ from the expression for N^{-1}. Note that the condition in the theorem statement: $\max_j \sum_i |m_{ij}| + \max_k |a_{k,k-1}| < 1$, is equivalent to $T + s < 1 \Rightarrow T/(1-s) < 1$ (observing that $s \in [0, 1)$).

$$\frac{|c_{ij}|}{\tilde{P}_{ij}} = \frac{|\sum_k \hat{n}_{ik} m_{kj}|}{\sum_k \hat{P}_{ik} P_{kj}} \leq \frac{\sum_k |\hat{n}_{ik}||m_{kj}|}{\sum_k \hat{P}_{ik} P_{kj}} = \frac{\sum_k |\hat{n}_{ik}||m_{kj}|}{\sum_k \left(|\hat{n}_{ik}|/\sum_l |\hat{n}_{lk}|\right)\left(|m_{kj}|/\sum_r |m_{rj}|\right)}$$

$$= \frac{(\sum_r |m_{rj}|)(\sum_k |\hat{n}_{ik}||m_{kj}|)}{\sum_k (|\hat{n}_{ik}||m_{kj}|)/(\sum_l |\hat{n}_{lk}|)} \leq \frac{\sum_r |m_{rj}| \sum_k |\hat{n}_{ik}||m_{kj}|}{\sum_k (|\hat{n}_{ik}||m_{kj}|/U)} \leq TU \frac{\sum_k |\hat{n}_{ik}||m_{kj}|}{\sum_k |\hat{n}_{ik}||m_{kj}|}$$

$$= TU \leq \frac{T}{(1-s)} < 1.$$

Please cite this article in press as: A. Srinivasan, Monte Carlo linear solvers with non-diagonal splitting, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.03.010

References

[1] J.H. Curtiss, Monte Carlo methods for the iteration of linear operators, Journal of Mathematical Physics 32 (1954) 209–232.
[2] J.H. Curtiss, 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, in: Symposium on Monte Carlo methods, University of Florida, 1954, John Wiley and Sons, New York, 1956, pp. 191–233.
[3] I.T. Dimov, Monte Carlo Methods for Applied Scientists, World Scientific, Singapore/London, 2008.
[4] I.T. Dimov, T.T. Dimov, T.V. Gurov, A new iterative Monte Carlo approach for inverse matrix problem, Journal of Computational and Applied Mathematics 92 (1998) 15–35.
[5] I.T. Dimov, A.N. Karaivanova, A power method with Monte Carlo iterations, in: Iliev, Kaschiev, Margenov, Sendov, Vassilevski (Eds.), Recent Advances in Numerical Methods and Appl. II, World Scientific, 1999, pp. 239–247.
[6] G.E. Forsythe, R.A. Leibler, Matrix inversion by a Monte Carlo method, Mathematical Tables and Other Aids to Computation 4 (1950) 127.
[7] J.H. Halton, Sequential Monte Carlo, Proceeding of the Cambridge Philosophical Society 58 Part 1 (1962) 57–78.
[8] M.T. Heath, Scientific Computing: An Introductory Survey, McGraw-Hill, New York, 1997.
[9] D.E. Knuth, The Art of Computer Programming, vol. 2: Seminumerical Algorithms, 3rd ed., Addison-Wesley, Reading, Massachusetts, 1998.
[10] G.A. Mikhailov, Optimization of weighted Monte Carlo Methods, Springer Series in Computational Physics, Springer-Verlag, Berlin, 1992.
[11] A. Srinivasan, Improved Monte Carlo linear solvers through non-diagonal splitting, Tech. Re TR-030203, Department of Computer Science, Florida State University (2003).
[12] A. Srinivasan, V. Aggarwal, Improved Monte Carlo linear solvers through non-diagonal splitting, in: Lecture Notes in Computer Science: Proceedings of the 2003 International Conference on Computational Science and its Applications, Springer-Verlag, 2003.
[13] A. Srinivasan, V. Aggarwal, Stochastic linear solvers, in: Proceedings of the SIAM Conference on Applied Linear Algebra, 2003, SIAM, 2003.
[14] A. Srinivasan, M. Mascagni, Monte Carlo techniques for estimating the Fiedler vector in graph applications, in: Lecture Notes in Computer Science, 2330, Springer-Verlag, 2002.
[15] W. Wasow, A note on the inversion of matrices by random walks, Mathematical Tables and Other Aids to Computation 6 (1952) 78.

Please cite this article in press as: A. Srinivasan, Monte Carlo linear solvers with non-diagonal splitting, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.03.010