

## Analysis of Monte Carlo accelerated iterative methods for sparse linear systems

Michele Benzi<sup>1</sup>, Thomas M. Evans<sup>2</sup>, Steven P. Hamilton<sup>3</sup>,  
Massimiliano Lupo Pasini<sup>4</sup>, and Stuart R. Slattery<sup>5</sup>

<sup>1</sup>*Department of Mathematics and Computer Science, Emory University, Atlanta, Georgia 30322, USA  
(benzi@mathcs.emory.edu).*

<sup>2</sup>*Oak Ridge National Laboratory, 1 Bethel Valley Rd., Oak Ridge, TN 37831, USA (evanstm@ornl.gov).*

<sup>3</sup>*Oak Ridge National Laboratory, 1 Bethel Valley Rd., Oak Ridge, TN 37831, USA (hamiltonsp@ornl.gov).*

<sup>4</sup>*Department of Mathematics and Computer Science, Emory University, Atlanta, Georgia 30322, USA  
(mlupopa@emory.edu).*

<sup>5</sup>*Oak Ridge National Laboratory, 1 Bethel Valley Rd., Oak Ridge, TN 37831, USA (slatterysr@ornl.gov).*

### SUMMARY

We consider hybrid deterministic-stochastic iterative algorithms for the solution of large, sparse linear systems. Starting from a convergent splitting of the coefficient matrix, we analyze various types of Monte Carlo acceleration schemes applied to the original preconditioned Richardson (stationary) iteration. These methods are expected to have considerable potential for resiliency to faults when implemented on massively parallel machines.

We establish sufficient conditions for the convergence of the hybrid schemes, and we investigate different types of preconditioners including sparse approximate inverses. Numerical experiments on linear systems arising from the discretization of partial differential equations are presented. Copyright © 2016 John Wiley & Sons, Ltd.

Received ...

KEY WORDS: sparse linear systems; iterative methods; Richardson iteration; preconditioning; Monte Carlo methods; sparse approximate inverses; resilience

### 1. INTRODUCTION

The next generation of computational science applications will require numerical solvers that are both reliable and capable of high performance on projected exascale platforms. In order to meet these goals, solvers must be resilient to soft and hard system failures, provide high concurrency on heterogeneous hardware configurations, and retain numerical accuracy and efficiency. In this paper we focus on the solution of large sparse systems of linear equations, for example of the kind arising from the discretization of partial differential equations (PDEs). A possible approach is to try to adapt existing solvers (such as preconditioned Krylov subspace or multigrid methods) to the new computational environments, and indeed several efforts are under way in this direction; see, e.g., [1, 14, 21, 25, 29] and references therein. An alternative approach is to investigate new algorithms that can address issues of resiliency, particularly fault tolerance and hard processor failures, as recently proposed in [13, 26]. In these methods, an underlying Monte Carlo Synthetic Acceleration Methods (MCSA), see [13, 26]. In these methods, an underlying

This is the author's final manuscript as accepted for publication, provided by the journal publisher. It has not been through the copyediting, typesetting, pagination and proofreading process, which may lead to differences between this version and the Version of Record. Please cite this article as doi: 10.1002/nla.2088

Contract/grant sponsor: Department of Energy (Office of Science); contract/grant number: ERKJ247

Copyright © 2016 John Wiley & Sons, Ltd.

Prepared using nlaauth.cls [Version: 2010/05/13 v2.00]

This article is protected by copyright. All rights reserved.

(deterministic) stationary Richardson iterative method is combined with a stochastic, Monte Carlo-based “acceleration” scheme. Ideally, the accelerated scheme will converge to the solution of the linear system in far fewer (outer) iterations than the basic scheme without Monte Carlo acceleration, with the added advantage that most of the computational effort is now relegated to the Monte Carlo portion of the algorithm, which is highly parallel and offers a more straightforward path to resiliency than standard, deterministic solvers. In addition, a careful combination of the Richardson and Monte Carlo parts of the algorithm allows to circumvent the well known problem of slow Monte Carlo error reduction; see [13].

Numerical evidence presented in [13] suggests that MCSA can be competitive, for certain classes of problems, with established deterministic solvers such as preconditioned conjugate gradients and GMRES. So far, however, no theoretical analysis of the convergence properties of these solvers has been carried out. In particular, it is not clear a priori whether the method, applied to a particular linear system, will converge. Indeed, the convergence of the underlying preconditioned Richardson iteration is not sufficient, in general, to guarantee the convergence of the MCSA-accelerated iteration. In other words, it is quite possible that the stochastic “acceleration” part of the algorithm may actually cause the hybrid method to diverge or stagnate.

In this paper we address this fundamental issue, discussing both necessary and sufficient conditions for convergence. We also discuss the choice of splitting, or preconditioner, and illustrate our findings by means of numerical experiments. We do not specifically consider in this paper the resiliency issue, which will be addressed elsewhere.

The paper is organized as follows. In Section 2 we provide an overview of existing Monte Carlo linear solver algorithms. In Section 3 we will discuss the convergence behavior of stochastic solvers, including a discussion of classes of matrices for which convergence can be guaranteed. Section 4 provides some numerical results illustrating properties of the various approaches and in Section 5 we give our conclusions.

## 2. STOCHASTIC LINEAR SOLVERS

Linear solvers based on stochastic techniques have a long history, going back to the famed 1928 paper by Courant, Friedrichs, and Lewy on finite difference schemes for PDEs [8]. Many authors have considered linear solvers based on Monte Carlo techniques, with important early contributions by Curtiss [9] and by Forsythe and Leibler [16]. More recent works include [2, 19], and [10], among others. Until recently, these methods have had mixed success at best, due to their generally inferior performance when compared to state-of-the-art deterministic solvers like multigrid or preconditioned Krylov methods. Current interest in resilient solvers, where some performance may be traded off for increased robustness in the presence of faults, has prompted a fresh look at methods incorporating Monte Carlo ideas [13, 27, 26].

As mentioned in [10], Monte Carlo methods may be divided into two broad classes: *direct methods*, such as those described in [10, 11], and *iterative methods*, which refer to techniques such as those presented in [18, 19]; see also [13, 27]. The first type consists of purely stochastic schemes, therefore the resulting error with respect to the exact solution is made of just a stochastic component. In contrast, the iterative Monte Carlo methods utilize more traditional iterative algorithms alongside the stochastic approach, generating two types of error: a *stochastic* one and a *systematic* one. In practice, it may be difficult to separate the two components; nevertheless, awareness of this intrinsic structure is useful, as it allows algorithm designers some flexibility in the choice of what part of the algorithm to target for refinement (e.g., trading off convergence speed for resilience by balancing the number of “deterministic” outer iterations against the number of random walks to be used within each iteration).

Consider a system of linear equations of the form

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

where  $A \in \mathbb{R}^{n \times n}$  and  $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$ . Equation (1) can be recast as a fixed point problem:

$$\mathbf{x} = H\mathbf{x} + \mathbf{f}, \quad (2)$$

where  $H = I - A$  and  $\mathbf{f} = \mathbf{b}$ . Assuming that the spectral radius  $\rho(H) < 1$ , the solution to (2) can be written in terms of a power series in  $H$  (Neumann series):

$$\mathbf{x} = \sum_{\ell=0}^{\infty} H^{\ell} \mathbf{f}.$$

Denoting the  $k$ th partial sum by  $\mathbf{x}^{(k)}$ , the sequence of approximate solutions  $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$  converges to the exact solution regardless of the initial guess  $\mathbf{x}_0$ .

By restricting the attention to a single component of  $\mathbf{x}$  we obtain

$$x_i = f_i + \sum_{\ell=1}^{\infty} \sum_{k_1=1}^n \sum_{k_2=1}^n \cdots \sum_{k_{\ell}=1}^n H_{i,k_1} H_{k_1,k_2} \cdots H_{k_{\ell-1},k_{\ell}} f_{k_{\ell}}. \quad (3)$$

The last equation can be reinterpreted as the realization of an estimator defined on a random walk. Let us start considering a random walk whose state space  $S$  is labeled by the set of indices of the forcing term  $\mathbf{f}$ :

$$S = \{1, 2, \dots, n\} \subset \mathbb{N}.$$

Each  $i$ th step of the random walk has a random variable  $k_i$  associated with it. The realization of  $k_i$  represents the index of the component of  $\mathbf{f}$  which is visited in the current step of the random walk. The construction of random walks is accomplished considering the directed graph associated with the matrix  $H$ . The nodes of this graph are labeled 1 through  $n$ , and there is a directed edge from node  $i$  to node  $j$  if and only if  $H_{i,j} \neq 0$ . Starting from a given node, the random walk consists of a sequence of nodes obtained by jumping from one node to the next along directed edges, choosing the next node at random according to a transition probability distribution matrix constructed from  $H$  or from  $H^T$ , see below. Note that it may happen that a row of  $H$  (or of  $H^T$ ) is all zero; this happens when there are no out-going (respectively, in-coming) edges from (respectively, to) the corresponding node. In this case, that particular random walk is terminated and another node is selected as the starting point for the next walk. The transition probabilities and the selection of the initial state of the random walk can be accomplished according to different modalities, leading to two distinct methods: the *forward* and *adjoint* methods. These methods are described next.

### 2.1. Forward method

Given a functional  $J$ , the goal is to compute its action on a vector  $\mathbf{x}$  by constructing a statistical estimator whose expected value equals  $J(\mathbf{x})$ . Each statistical sampling is represented by a random walk and it contributes to build an estimate of the expected value. Towards this goal, it is necessary to introduce an initial probability distribution and a transition matrix so that random walks are well defined. Recalling Riesz's representation theorem one can write

$$J(\mathbf{x}) = \langle \mathbf{h}, \mathbf{x} \rangle = \sum_{i=1}^n h_i x_i,$$

where  $\mathbf{h} \in \mathbb{R}^n$  is the Riesz representative in  $\mathbb{R}^n$  of the functional  $J$ . Such a representative can be used to build the initial probability  $\tilde{p} : S \rightarrow [0, 1]$  of the random walk as

$$\tilde{p}(k_0 = i) = \tilde{p}_{k_0} = \frac{|h_i|}{\sum_{i=1}^n |h_i|}.$$

It is important to highlight that the role of vector  $\mathbf{h}$  is confined to the construction of the initial probability, and that  $\mathbf{h}$  is not used afterwards in the stochastic process. A possible choice for the

transition probability  $P$  can be

$$p(k_\ell = j \mid k_{\ell-1} = i) = P_{ij} = \frac{|H_{ij}|}{\sum_{k=1}^n |H_{ik}|}$$

where

$$p(\cdot, i) : S \rightarrow [0, 1], \quad \forall i \in S$$

and  $k_\ell \in S$  represents the state reached at a generic  $\ell$ th step of the random walk. A related sequence of random variables  $w_{ij}$  can be defined as

$$w_{ij} = \frac{H_{ij}}{P_{ij}}$$

The probability distribution of the random variables  $w_{ij}$  is represented by the transition matrix that governs the stochastic process. The  $w_{ij}$  quantities just introduced can be used to build one more sequence of random variables. At first we introduce quantities  $W_\ell$

$$W_0 = \frac{h_{k_0}}{\tilde{p}_{k_0}}, \quad W_\ell = W_{\ell-1} w_{k_{\ell-1}, k_\ell}$$

In probability theory, a random walk is itself envisioned as a random variable that can assume multiple values consisting of different realizations. Indeed, given a starting point, there are in general many choices (one for each nonzero in the corresponding row of  $P$ ) to select the second state and from there on recursively. The actual feasibility of a path and the frequency with which it is selected depend on the initial probability and on the transition matrix. By introducing  $\nu$  as a realization of a random walk, we define

$$X(\nu) = \sum_{\ell=0}^{\infty} W_\ell f_{k_\ell}$$

as the random variable associated with a specific realization  $\nu$ . We can thus define the estimator  $\theta$  as

$$\theta = E[X] = \sum_{\nu} P_\nu X(\nu),$$

where  $\nu$  ranges over all possible realizations.  $P_\nu$  is the probability associated with a specific realization of the random walk. It can be proved (see [18] and [19]) that

$$E[W_\ell f_{k_\ell}] = \langle \mathbf{h}, H^\ell \mathbf{f} \rangle, \quad \ell = 0, 1, 2, \dots$$

and

$$\theta = E \left[ \sum_{\ell=0}^{\infty} W_\ell f_{k_\ell} \right] = \langle \mathbf{h}, \mathbf{x} \rangle.$$

A possible choice for  $\mathbf{h}$  is a vector of the standard basis,  $\mathbf{h} = \mathbf{e}_i$ . This would correspond to setting the related initial probability to a Kronecker delta:

$$\tilde{p}(k_0 = j) = \delta_{ij}.$$

By doing so, we have  $k_0 = i$  and

$$\theta = x_i = f_i + \sum_{l=1}^{\infty} \sum_{k_1=1}^n \sum_{k_2=1}^n \cdots \sum_{k_\ell=1}^n P_{i,k_1} P_{k_1,k_2} \cdots P_{k_{\ell-1},k_\ell} w_{i,k_1} w_{k_1,k_2} \cdots w_{k_{\ell-1},k_\ell} f_{k_\ell}. \quad (4)$$

As regards the variance, we recall the following relation:

$$\text{Var} \left[ \sum_{\ell=0}^{\infty} W_\ell f_{k_\ell} \right] = E \left[ \sum_{\ell=0}^{\infty} W_\ell^2 f_{k_\ell}^2 \right] - \left( E \left[ \sum_{\ell=0}^{\infty} W_\ell f_{k_\ell} \right] \right)^2. \quad (5)$$

Hence, the variance can be computed as the difference between the second moment of the random variable and the square of its first moment.

In order to apply the Central Limit Theorem (CLT) to the estimators defined above, we must require that the estimators have both finite expected value and finite variance. This is equivalent to checking the finiteness of the expected value and second moment. Therefore, we have to impose the following conditions:

$$E \left[ \sum_{\ell=0}^{\infty} W_{\ell} f_{k_{\ell}} \right] < \infty, \quad (6)$$

$$E \left[ \sum_{\ell=0}^{\infty} W_{\ell}^2 f_{k_{\ell}}^2 \right] < \infty. \quad (7)$$

The forward method presented above, however, has the limitation of employing an entire set of realizations to estimate just a single entry of the solution at a time. Hence, in order to estimate the entire solution vector for Eq. (1), we have to employ a separate set of realizations for each entry in the solution vector. This limitation can be circumvented by the adjoint method which we describe below.

*Remark 1*

It is important to note that in order to construct the random walks, access to the individual entries of  $H$  is required. Hence,  $H$  needs to be formed explicitly and therefore must be sparse in order to have a practical algorithm.

2.2. *Adjoint method*

A second Monte Carlo method can be derived by considering the linear system adjoint to (1):

$$A^T \mathbf{y} = \mathbf{d}, \quad (8)$$

where  $\mathbf{y}$  and  $\mathbf{d}$  are the adjoint solution and source term. Equation (8) can be recast in a manner similar to (2):

$$\mathbf{y} = H^T \mathbf{y} + \mathbf{d}.$$

Note that  $\rho(H^T) = \rho(H) < 1$ , hence convergence of the Neumann series (fixed point iteration) for (1) guarantees convergence for the adjoint system (8).

Exploiting the following inner product equivalence:

$$\langle A^T \mathbf{y}, \mathbf{x} \rangle = \langle \mathbf{y}, A \mathbf{x} \rangle,$$

it follows that

$$\langle \mathbf{x}, \mathbf{d} \rangle = \langle \mathbf{y}, \mathbf{f} \rangle. \quad (9)$$

By writing the Neumann series for the solution to (8) we have:

$$\mathbf{y} = \sum_{\ell=0}^{\infty} (H^T)^{\ell} \mathbf{d},$$

and focusing on a single entry of the solution vector:

$$y_i = d_i + \sum_{\ell=1}^{\infty} \sum_{k_1=1}^n \sum_{k_2=1}^n \cdots \sum_{k_{\ell}=1}^n H_{i,k_1}^T H_{k_1,k_2}^T \cdots H_{k_{\ell-1},k_{\ell}}^T d_{k_{\ell}}.$$

The undetermined quantities in the dual problem (8) are  $\mathbf{y}$  and  $\mathbf{d}$ . Therefore, two constraints are required: the first constraint is Eq. (9) and as a second constraint we select  $\mathbf{d}$  to be one of the standard

basis vectors. Applying this choice of  $\mathbf{d}$  to (9) we get:

$$\langle \mathbf{y}, \mathbf{f} \rangle = \langle \mathbf{x}, \mathbf{d} \rangle = x_i.$$

In order to give a stochastic interpretation of the adjoint method similar to the one obtained for the forward method, we introduce the initial probability:

$$\tilde{p}(k_0 = i) = \frac{|f_i|}{\|\mathbf{f}\|_1}$$

and the initial weight:

$$W_0 = \|\mathbf{f}\|_1 \frac{f_i}{|f_i|}.$$

The transition probability is defined as

$$p(k_\ell = j | k_{\ell-1} = i) = P_{ij} = \frac{|H_{ij}^T|}{\sum_{k=1}^n |H_{ik}^T|} = \frac{|H_{ji}|}{\sum_{k=1}^n |H_{ki}|}$$

and the sequence of weights as follows:

$$w_{ij} = \frac{H_{ji}}{P_{ij}}.$$

By reformulating the fixed point scheme in its statistical interpretation, the following formula holds for the estimator of the solution vector associated with the adjoint method: it is the vector  $\boldsymbol{\theta} \in \mathbb{R}^n$  such that

$$\begin{aligned} \theta_i &= E \left[ \sum_{\ell=0}^{\infty} W_\ell \delta_{k_\ell, i} \right] \\ &= \sum_{\ell=0}^{\infty} \sum_{k_0=1}^n \sum_{k_1=1}^n \sum_{k_2=1}^n \cdots \sum_{k_\ell=1}^n f_{k_0} P_{k_0, k_1} P_{k_1, k_2} \cdots P_{k_{\ell-1}, k_\ell} w_{k_0, k_1} \cdots w_{k_{\ell-1}, k_\ell} \delta_{k_\ell, i}. \end{aligned} \quad (10)$$

This estimator is known in literature as *collision estimator*.

The forward method adds a contribution to the component of the solution vector where the random walk began, based on the value of the source vector in the state in which the walk currently resides. The adjoint method, on the other hand, adds a contribution to the component of the solution vector where the random walk currently resides based on the value of the source vector in the state in which the walk began. The Kronecker delta at the end of the series (10) represents a filter, indicating that only a subset of realizations contribute to the  $j$ th component of the solution vector.

The variance is given by

$$\text{Var} \left[ \sum_{\ell=0}^{\infty} W_\ell \delta_{k_\ell, i} \right] = E \left[ \sum_{\ell=0}^{\infty} W_\ell^2 \delta_{k_\ell, i} \right] - \left( E \left[ \sum_{\ell=0}^{\infty} W_\ell \delta_{k_\ell, i} \right] \right)^2, \quad i = 1, \dots, n. \quad (11)$$

Along the same lines as the development for the forward method, we must impose finiteness of the expected value and second moment. Therefore, the following conditions must be verified:

$$E \left[ \sum_{\ell=0}^{\infty} W_\ell \delta_{k_\ell, i} \right] < \infty \quad i = 1, \dots, n \quad (12)$$

and

$$E \left[ \sum_{\ell=0}^{\infty} W_\ell^2 \delta_{k_\ell, i} \right] < \infty, \quad i = 1, \dots, n. \quad (13)$$

The main advantage of this method, compared to the forward one, consists in the fact that a single set of realizations is used to estimate the entire solution vector. Unless only a small portion

of the problem is of interest, this property often leads to the adjoint method being favored over the forward method. In other terms, the adjoint method should be preferred when approximating the solution globally over the entire computational domain, while the forward method is especially useful when approximating the solution locally.

In literature another estimator is employed along with the adjoint Monte Carlo method, the so called *expected value* estimator. Its formulation is as follows: it is the vector  $\theta \in \mathbb{R}^n$  such that

$$\begin{aligned} \theta_i &= E \left[ f_i + \sum_{\ell=0}^{\infty} W_{\ell} H_{k_{\ell},i}^T \right] \\ &= f_i + \sum_{\ell=0}^{\infty} \sum_{k_0=1}^n \sum_{k_1=1}^n \sum_{k_2=1}^n \cdots \sum_{k_{\ell}=1}^n f_{k_0} P_{k_0,k_1} P_{k_1,k_2} \cdots P_{k_{\ell-1},k_{\ell}} w_{k_0,k_1} \cdots w_{k_{\ell-1},k_{\ell}} H_{k_{\ell},i}^T. \end{aligned} \quad (14)$$

Hence, the expected value estimator averages the deterministic contribution of the iteration matrix over all the potential states  $j$  that might be reached from the current state  $\ell$ . The variance in this case becomes:

$$\text{Var} \left[ \sum_{\ell=0}^{\infty} W_{\ell} H_{k_{\ell},i}^T \right] = E \left[ \sum_{\ell=0}^{\infty} W_{\ell}^2 H_{k_{\ell},i}^T \right] - \left( E \left[ \sum_{\ell=0}^{\infty} W_{\ell} H_{k_{\ell},i}^T \right] \right)^2, \quad i = 1, \dots, n. \quad (15)$$

### 2.3. Hybrid stochastic/deterministic methods

The direct methods described in Sections 2.1 and 2.2 suffer from a slow rate of convergence due to the  $\frac{1}{\sqrt{N}}$  behavior dictated by the central limit theorem ( $N$  here is the number of random walks used to estimate the solution). Furthermore, when the spectral radius of the iteration matrix is close to unity, each individual random walk may require a large number of transitions to approximate the corresponding components in the Neumann series. To offset the slow convergence of the central limit theorem, schemes have been proposed which combine traditional fixed point iterative methods with the stochastic solvers. The first such method, due to Halton, was termed the Sequential Monte Carlo (SMC) method, and can be written as follows.

---

#### Algorithm 1: Sequential Monte Carlo

---

**Data:**  $H, \mathbf{b}, \mathbf{x}_0$   
**Result:**  $\mathbf{x}_{num}$

```

1  $l = 0$ ;
2 while not reached convergence do
3    $\mathbf{r}^l = \mathbf{b} - A\mathbf{x}^l$ ;
4    $\delta\mathbf{x}^{l+1} \approx (I - H)^{-1}\mathbf{r}^l$ ;    % Computed via Standard MC  $\mathbf{x}^{l+1} = \mathbf{x}^l + \delta\mathbf{x}^{l+1}$ ;
5    $l = l + 1$ ;
6 end
7  $\mathbf{x}_{num} = \mathbf{x}^{l+1}$ ;
```

---

The Monte Carlo linear solver method is used to compute the update  $\delta\mathbf{x}^l$ . This algorithm is equivalent to a Richardson iteration accelerated by a correction obtained by approximately solving the error-residual equation

$$(I - H)\delta\mathbf{x}^{l+1} = \mathbf{r}^l. \quad (16)$$

If this equation were to be solved exactly, the corresponding approximation  $\mathbf{x}^{l+1} = \mathbf{x}^l + \delta\mathbf{x}^{l+1}$  would be the exact solution to the linear system. This is of course impractical, since solving (16) is equivalent to solving the original linear system (1). Instead, the correction is obtained by solving (16) only approximately, using a Monte Carlo method. Because Monte Carlo is only applied within a single iteration, the central limit theorem is only applicable within that iteration rather than to the overall convergence behavior of the algorithm. This allows a trade-off between the amount of time and effort spent on the inner (stochastic) and outer (deterministic) iterations, which can take account the competing goal of reliability and rapid convergence.

A further extension of Halton's method, termed *Monte Carlo Synthetic Acceleration* (MCSA), has been recently introduced in [27] and [13]. The MCSA algorithm can be written as:

---

**Algorithm 2:** Monte Carlo Synthetic Acceleration

---

**Data:**  $H, \mathbf{b}, \mathbf{x}_0$   
**Result:**  $\mathbf{x}_{num}$

```

1  $l = 0;$ 
2 while not reached convergence do
3    $\mathbf{x}^{l+\frac{1}{2}} = H\mathbf{x}^l + \mathbf{b};$ 
4    $\mathbf{r}^{l+\frac{1}{2}} = \mathbf{b} - A\mathbf{x}^{l+\frac{1}{2}};$ 
5    $\delta\mathbf{x}^{l+\frac{1}{2}} \approx (I - H)^{-1}\mathbf{r}^{l+\frac{1}{2}};$     % Computed via Standard MC  $\mathbf{x}^{l+1} = \mathbf{x}^{l+\frac{1}{2}} + \delta\mathbf{x}^{l+\frac{1}{2}};$ 
6    $l = l + 1;$ 
7 end
8  $\mathbf{x}_{num} = \mathbf{x}^{l+1};$ 

```

---

As with SMC, a Monte Carlo linear solver is used to compute the updating contribution  $\delta\mathbf{x}^{l+\frac{1}{2}}$ . In this approach, an extra step of Richardson iteration is added to smooth out some of the high-frequency noise introduced by the Monte Carlo process. This way, the deterministic and stochastic components of the algorithm act in a complementary fashion.

Obviously, a minimum requirement is that the linear system can be written in the form (2) with  $\rho(H) < 1$ . This is typically achieved by preconditioning. That is, we find an invertible matrix  $P$  such that  $H = I - P^{-1}A$  satisfies  $\rho(H) < 1$ , and we apply the method to the fixed point problem (2) where  $H = I - P^{-1}A$  and  $\mathbf{f} = P^{-1}\mathbf{b}$ . In other words, the underlying deterministic iteration is a preconditioned Richardson iteration. Various choices of the preconditioner are possible; a detailed discussion of this issue is deferred until Section 3.5. Here we note only that because we need explicit knowledge of the entries of  $H$ , not all preconditioning choices are viable; in particular,  $P$  needs to be such that  $H = I - P^{-1}A$  retains a high degree of sparsity. Unless otherwise specified, below we assume that the transformation of the original linear system (1) to the fixed point form (2) with  $\rho(H) < 1$  has already been carried out.

### 3. CONVERGENCE BEHAVIOR OF STOCHASTIC METHODS

Interestingly, the convergence requirements imposed by the Monte Carlo estimator and the corresponding variance can be reformulated in a purely deterministic setting. For instance, the condition of finiteness of the expected value turns out to be equivalent to requiring

$$\rho(H) < 1, \quad (17)$$

where  $H$  is the iteration matrix of the fixed point scheme. Indeed, we can see from (4) and (10) that the expected value is expressed in terms of power series of  $H$ , and the condition  $\rho(H) < 1$  is a necessary and sufficient condition for the Neumann series to converge.

Next, we address the finiteness requirement for the second moment. Equations (5) and (11) for the forward and the adjoint method, respectively, show that the second moment can be reinterpreted as a power series with respect to the matrices defined as follows:

$$\hat{H}_{ij} = \frac{H_{ij}^2}{P_{ij}} \quad \text{- Forward Method}$$

and

$$\hat{H}_{ij} = \frac{H_{ji}^2}{P_{ij}} \quad \text{- Adjoint Method.}$$

In order for the corresponding power series to converge, we must require

$$\rho(\hat{H}) < 1. \quad (18)$$

Hence, condition (17) is required for a generic fixed point scheme to reach convergence, whereas the extra condition (18) is typical of the stochastic schemes studied in this work. Moreover, since the finiteness of the variance automatically entails the finiteness of the expected value, we can state that (18) implicitly entails (17), whereas the converse is not true in general.

### 3.1. Necessary and sufficient conditions

Here we report some results presented in [28] and [20], concerning necessary conditions and sufficient conditions for convergence. In particular, these papers discuss suitable choices for constructing the transition probability matrix,  $P$ .

The construction of the transition probability must obviously satisfy the following constraints (called *transition conditions*):

$$\begin{cases} P_{ij} \geq 0 \\ \sum_{j=1}^N P_{ij} = 1. \end{cases}$$

One additional requirement relates the sparsity pattern of  $H$  to that of the transition probabilities:

$$\text{Forward Method: } H_{ij} \neq 0 \Rightarrow P_{ij} \neq 0,$$

$$\text{Adjoint Method: } H_{ji} \neq 0 \Rightarrow P_{ij} \neq 0.$$

The following auxiliary result can be found in [20].

#### Lemma 1

Consider a generic vector  $\mathbf{g} = (g_1, g_2, \dots, g_N)^T$  where at least one element is nonzero,  $g_k \neq 0$  for some  $k \in \{1, \dots, N\}$ . Then, the following statements hold:

- (i) for any probability distribution vector  $\beta = (\beta_1, \beta_2, \dots, \beta_N)^T$  satisfying the transition conditions,  $\sum_{k=1}^N \frac{g_k^2}{\beta_k} \geq \left( \sum_{k=1}^N |g_k| \right)^2$ ; moreover, the lower bound is attained for the probability vector defined by  $\beta_k = \frac{|g_k|}{\sum_{k=1}^N |g_k|}$ ;
- (ii) there always exists a probability vector  $\beta$  such that  $\sum_{k=1}^N \frac{g_k^2}{\beta_k} \geq c$ , for all  $c > 1$ .

Consider now a generic realization of a random walk, truncated at a certain  $k$ th step:

$$\nu_k : r_0 \rightarrow r_1 \rightarrow r_2 \rightarrow \dots \rightarrow r_k,$$

and the corresponding statistical estimator associated with the Forward Monte Carlo method:

$$X(\nu_k) = \frac{H_{r_0, r_1} H_{r_1, r_2} \cdots H_{r_{k-1}, r_k}}{P_{r_0, r_1} P_{r_1, r_2} \cdots P_{r_{k-1}, r_k}} f_{r_k}.$$

Then, the following result holds (see [20]).

#### Theorem 1

(Forward method version) Let  $H \in \mathbb{R}^{n \times n}$  be such that  $\|H\|_\infty < 1$ . Consider  $\nu_k$  as the realization of a random walk  $\nu$  truncated at the  $k$ th step. Then, there always exists a transition matrix  $P$  such that  $\text{Var}(X(\nu_k)) \rightarrow 0$  and  $\text{Var}(\sum_\nu X(\nu_k))$  is bounded as  $k \rightarrow \infty$ .

If we introduce the estimator associated with the Adjoint Monte Carlo:

$$X(\nu_k) = \frac{H_{r_0, r_1}^T H_{r_1, r_2}^T \cdots H_{r_{k-1}, r_k}^T}{P_{r_0, r_1} P_{r_1, r_2} \cdots P_{r_{k-1}, r_k}} \text{sign}(f_{r_0}) \|\mathbf{f}\|_1,$$

then we can state a theorem analogous to 1 (see [20]).

### Theorem 2

(Adjoint method version) Let  $H \in \mathbb{R}^{n \times n}$  with  $\|H\|_1 < 1$ . Consider  $\nu_k$  as the realization of a random walk  $\nu$  truncated at the  $k$ th step. Then, there always exists a transition matrix  $P$  such that  $\text{Var}(X(\nu_k)) \rightarrow 0$  and  $\text{Var}(\sum_{\nu} X(\nu_k))$  is bounded as  $k \rightarrow \infty$ .

These results represent sufficient (but not necessary) conditions for the convergence of the forward and adjoint Monte Carlo and can be easily verified if  $H$  is explicitly available. However, in many cases the conditions  $\|H\|_{\infty} < 1$  or  $\|H\|_1 < 1$  may be too restrictive.

The connection between Lemma 1 and Theorems 1-2 will be explained in the next section, dedicated to the definition of transition probabilities.

### 3.2. Construction of transition probabilities

The way the transition probability is defined has a significant impact on the properties of the resulting algorithm, and in many circumstances the choice can make the difference between convergence or divergence of the stochastic scheme. Two main approaches have been considered in the literature: *uniform probabilities* and *weighted probabilities*. We discuss these next.

3.2.1. *Uniform probabilities* With this approach, the transition matrix  $P$  is such that all the transitions corresponding to each row have equal probability of occurring:

$$\begin{aligned} \text{Forward : } P_{ij} &= \begin{cases} 0 & \text{if } H_{ij} = 0, \\ \frac{1}{\#(\text{non-zeros in row } i \text{ of } H)} & \text{if } H_{ij} \neq 0; \end{cases} \\ \text{Adjoint : } P_{ij} &= \begin{cases} 0 & \text{if } H_{ji} = 0, \\ \frac{1}{\#(\text{non-zeros in column } i \text{ of } H)} & \text{if } H_{ji} \neq 0. \end{cases} \end{aligned}$$

The Monte Carlo approach resorting to this definition of the transition matrix, in accordance to [2], is called *Uniform Monte Carlo* (UM).

3.2.2. *Weighted probabilities* An alternative definition of transition matrices aims to associate nonzero probability to the nonzero entries of  $H$  accordingly to their magnitude. For instance, we may employ the following definition:

$$\text{Forward : } p(k_i = j \mid k_{i-1} = i) = P_{ij} = \frac{|H_{ij}|^p}{\sum_{k=1}^n |H_{ik}|^p},$$

and

$$\text{Adjoint : } p(k_i = j \mid k_{i-1} = i) = P_{ij} = \frac{|H_{ji}|^p}{\sum_{k=1}^n |H_{ki}|^p},$$

where  $p \in \mathbb{N}$ . The case  $p = 1$  is called *Monte Carlo Almost Optimal* (MAO). The reason for the ‘‘almost optimal’’ designation can be understood looking at Lemma 1, as the quantity  $\sum_{k=1}^N \frac{g_k^2}{\beta_k}$  is minimized when the probability vector is defined as  $\beta_k = \frac{|g_k|}{\sum_{k=1}^N |g_k|}$ . Indeed, Lemma 1 implies that

the almost optimal probability minimizes the  $\infty$ -norm of  $\hat{H}$  for the forward method and the 1-norm of  $\hat{H}$  for the adjoint method, since the role of  $\mathbf{g}$  in Lemma 1 is played by the rows of  $H$  in the former case and by the columns of  $H$  in the latter one. This observation provides us with easily computable upper bounds for  $\rho(\hat{H})$ .

### 3.3. Classes of matrices with guaranteed convergence

On the one hand, sufficient conditions for convergence of Monte Carlo linear solvers are very restrictive; see, e.g., [28] and [20]. On the other hand, the necessary and sufficient condition in [20] requires knowledge of  $\rho(\hat{H})$ , which is not readily available. Note that explicit computation of

$\rho(\hat{H})$  is quite expensive, comparable to the cost of solving the original linear system. While ensuring that  $\rho(H) < 1$  (by means with appropriate preconditioning) is in many cases possible, guaranteeing that  $\rho(\hat{H}) < 1$  is generally much more problematic.

Here we identify matrix types for which both conditions can be satisfied by an appropriate choice of splitting, so that convergence of the Monte Carlo scheme is guaranteed.

**3.3.1. SDD matrices** One of these categories is represented by strictly diagonally dominant (SDD) matrices. We investigate under which conditions diagonal preconditioning is enough to ensure convergence. We recall the following definitions.

*Definition 1*

A matrix  $A \in \mathbb{R}^{n \times n}$  is strictly diagonally dominant by rows if

$$|a_{ij}| > \sum_{\substack{i=1 \\ i \neq j}}^n |a_{ij}|. \quad (19)$$

*Definition 2*

A matrix  $A \in \mathbb{R}^{n \times n}$  is strictly diagonally dominant by columns if  $A^T$  is strictly diagonally dominant by rows, i.e.,

$$|a_{ij}| > \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|. \quad (20)$$

Suppose  $A$  is SDD by rows. Then we can apply left diagonal (Jacobi) preconditioning, obtaining an iteration matrix  $H = I - P^{-1}A$  such that  $\|H\|_\infty < 1$ . Introducing a MAO transition probability for the forward method:

$$P_{ij} = \frac{|H_{ij}|}{\sum_{k=1}^n |H_{ik}|}$$

we have that the entries of  $\hat{H}$  are defined as follows:

$$\hat{H}_{ij} = \frac{H_{ij}^2}{P_{ij}} = |H_{ij}| \left( \sum_{k=1}^n |H_{ik}| \right).$$

Consequently,

$$\sum_{j=1}^n |\hat{H}_{ij}| = \sum_{j=1}^n \hat{H}_{ij} = \left( \sum_{j=1}^n |H_{ij}| \right) \left( \sum_{k=1}^n |H_{ik}| \right) = \left( \sum_{j=1}^n |H_{ij}| \right)^2 < 1, \quad \forall i = 1, \dots, n.$$

This implies that  $\rho(\hat{H}) \leq \|\hat{H}\|_\infty < 1$ , guaranteeing the forward Monte Carlo converges. However, nothing can be said *a priori* about the convergence of the adjoint method.

On the other hand, if (20) holds, we can apply right diagonal (Jacobi) preconditioning, which results in an iteration matrix  $H = I - AP^{-1}$  such that  $\|H\|_1 < 1$ . In this case, by a similar reasoning we conclude that the adjoint method converges, owing to  $\|\hat{H}\|_1 < 1$ ; however, nothing can be said *a priori* about the forward method.

Finally, it is clear that if  $A$  is SDD by rows *and* by columns, then a (left or right) diagonal preconditioning will result in the convergence of both the forward and the adjoint Monte Carlo schemes.

**3.3.2. GDD matrices** Another class of matrices for which the convergence of MC solvers is ensured is that of generalized diagonally dominant (GDD) matrices. We recall the following definition.

*Definition 3*

A square matrix  $A \in \mathbb{C}^{n \times n}$  is said to be *generalized diagonally dominant* if

$$|a_{ii}|d_i \geq \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|d_j, \quad i = 1, \dots, n,$$

for some positive vector  $\mathbf{d} = (d_1, \dots, d_n)^T$ .

Note that this means that there exists a diagonal matrix  $\Delta$  such that  $A\Delta$  is SDD. A proper subclass of the class of GDD matrices is represented by the nonsingular  $M$ -matrices. Recall that  $A$  is a nonsingular  $M$ -matrix if it is of the form  $A = rI - B$  where  $B$  is nonnegative and  $r > \rho(B)$ . It can be shown (see, e.g., [7]) that a matrix  $A \in \mathbb{R}^{n \times n}$  is a nonsingular  $M$ -matrix if and only if there exists a positive diagonal matrix  $\Delta$  such that  $A\Delta$  is SDD by rows. Clearly, every nonsingular  $M$ -matrix is GDD.

It is well known (see, e.g., [3]) that the classical Jacobi, Block Jacobi and Gauss–Seidel splittings are convergent if  $A$  is a nonsingular  $M$ -matrix. However, this is not enough to ensure the convergence of MC schemes based on the corresponding fixed point (preconditioned Richardson) iteration, since in general we cannot expect that  $\rho(\hat{H}) < 1$ .

Nevertheless, if  $A$  is an  $M$ -matrix there exist efficient methods to determine a diagonal scaling of  $A$  so that the resulting matrix is SDD by rows. Note that the scaled matrix is still an  $M$ -matrix, therefore applying left Jacobi preconditioning to this matrix will guarantee that both  $\rho(H) < 1$  and  $\rho(\hat{H}) < 1$ .

In [23], the author presents a procedure to determine whether a given matrix  $A \in \mathbb{C}^{n \times n}$  is GDD (in which case the diagonal scaling that makes  $A$  GDD is produced), or not. The algorithm can be described as follows.

---

**Algorithm 3:** Algorithm to determine whether a matrix is GDD

---

**Data:** matrix  $A$ ,  $a_{ii} \neq 0$ ,  $i = 1, \dots, n$   
**Result:**  $d_i$ ,  $i = 1, \dots, n$

- 1 Compute  $S_i = \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|$ ,  $i = 1, \dots, n$
- 2 Set  $t = 0$
- 3 **for**  $i = 1, \dots, n$  **do**
- 4     **if**  $|a_{ii}| > S_i$  **then**
- 5          $t = t + 1$
- 6     **end**
- 7 **end**
- 8 **if**  $t = 0$  **then**
- 9     print “A is not GDD”
- 10 **else if**  $t = n$  **then**
- 11     print “A is GDD”
- 12 **else**
- 13     **for**  $i = 1, \dots, n$  **do**
- 14          $d_i = \frac{S_i + \varepsilon}{|a_{ii}| + \varepsilon}$   $\varepsilon > 0$ ,  $j = 1, \dots, n$
- 15          $a_{ji} = a_{ji} \cdot d_i$
- 16     **end**
- 17     Go to step 1
- 18 **end**

---

This procedure, which in practice converges very fast, turns a generalized diagonally dominant matrix (in particular, a nonsingular  $M$ -matrix) into a strictly diagonally dominant matrix by rows.

By replacing  $S_i = \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|$  at step 1 with  $S_j = \sum_{\substack{i=1 \\ i \neq j}}^n |a_{ij}|$  and by replacing  $a_{ji} = a_{ji} \cdot d_i$  with  $a_{ji} = a_{ji} \cdot d_j$ , we obtain the algorithm that turns a GDD matrix into a matrix that is SDD by columns.

Once we have applied this transformation to the original matrix  $A$  the Monte Carlo scheme combined with diagonal preconditioning is ensured to converge.

3.3.3. *Block diagonally dominant matrices* In this section we analyze situations in which block diagonal preconditioning can produce a convergent Monte Carlo linear solver. Assume that  $A$  has been partitioned into  $p \times p$  block form, and that each diagonal block has size  $n_i$  with  $n_1 + \dots + n_p = n$ . Assume further that each diagonal block  $A_{ii}$  is nonsingular. The iteration matrix  $H \in \mathbb{R}^{n \times n}$  resulting from a block diagonal left preconditioning is

$$H = \begin{bmatrix} 0_{n_1 \times n_1} & -A_{11}^{-1}A_{12} & \cdots & \cdots & -A_{11}^{-1}A_{1p} \\ -A_{22}^{-1}A_{21} & 0_{n_2 \times n_2} & -A_{22}^{-1}A_{23} & \cdots & -A_{22}^{-1}A_{2p} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -A_{pp}^{-1}A_{p1} & \cdots & \cdots & -A_{pp}^{-1}A_{p,p-1} & 0_{n_p \times n_p} \end{bmatrix}.$$

Below, we denote with “ $i|m$ ” the modulo operation applied to the integers  $i$  and  $m$ . The symbol “ $\lfloor \cdot \rfloor$ ” stands for the floor function, as usual.

Consider first the forward method. Assuming (for ease of notation) that all the blocks have the same size  $m = n/p$ , the entries of the MAO transition probability matrix become:

$$P_{ij} = \frac{|H_{ij}|}{\sum_{k=1}^n |H_{ik}|} = \frac{\left| \left( [A_{\lfloor \frac{i}{m} \rfloor \lfloor \frac{j}{m} \rfloor}]^{-1} A_{\lfloor \frac{i}{m} \rfloor \lfloor \frac{j}{m} \rfloor} \right)_{i|m, j|m} \right|}{\sum_{\substack{k=1 \\ k \neq i}}^n \left| \left( [A_{\lfloor \frac{i}{m} \rfloor \lfloor \frac{k}{m} \rfloor}]^{-1} A_{\lfloor \frac{i}{m} \rfloor \lfloor \frac{k}{m} \rfloor} \right)_{i|m, k|m} \right|}.$$

Consequently, the entries of  $\hat{H}$  are given by

$$\begin{aligned} \hat{H}_{ij} &= |H_{ij}| \left( \sum_{k=1}^n |H_{ik}| \right) \\ &= \left| \left( [A_{\lfloor \frac{i}{m} \rfloor \lfloor \frac{j}{m} \rfloor}]^{-1} A_{\lfloor \frac{i}{m} \rfloor \lfloor \frac{j}{m} \rfloor} \right)_{i|m, j|m} \right| \sum_{\substack{k=1 \\ k \neq i}}^n \left| \left( [A_{\lfloor \frac{i}{m} \rfloor \lfloor \frac{k}{m} \rfloor}]^{-1} A_{\lfloor \frac{i}{m} \rfloor \lfloor \frac{k}{m} \rfloor} \right)_{i|m, k|m} \right|. \end{aligned}$$

Computing the sum over a generic row of  $\hat{H}$ , we obtain

$$\sum_{j=1}^n |\hat{H}_{ij}| = \sum_{j=1}^n \hat{H}_{ij} = \left( \sum_{\substack{j=1 \\ j \neq i}}^n \left| \left( [A_{\lfloor \frac{i}{m} \rfloor \lfloor \frac{j}{m} \rfloor}]^{-1} A_{\lfloor \frac{i}{m} \rfloor \lfloor \frac{j}{m} \rfloor} \right)_{i|m, j|m} \right| \right)^2.$$

Consider now the quantity  $\|\hat{H}\|_\infty$ . Clearly,

$$\|\hat{H}\|_\infty < 1 \Leftrightarrow \sum_{\substack{j=1 \\ j \neq i}}^n \left| \left( [A_{\lfloor \frac{i}{m} \rfloor \lfloor \frac{j}{m} \rfloor}]^{-1} A_{\lfloor \frac{i}{m} \rfloor \lfloor \frac{j}{m} \rfloor} \right)_{i|m, j|m} \right| < 1, \quad \forall i = 1, \dots, n.$$

A sufficient condition for this to happen is that

$$\sum_{\substack{k=1 \\ k \neq h}}^p \|A_{hh}^{-1} A_{hk}\|_\infty < 1, \quad \forall h = 1, \dots, p. \quad (21)$$

Introducing the matrix  $\tilde{H} \in \mathbb{R}^{p \times p}$  defined as

$$\tilde{H} = \begin{bmatrix} 0 & \|A_{11}^{-1}A_{12}\|_\infty & \cdots & \cdots & \|A_{11}^{-1}A_{1p}\|_\infty \\ \|A_{22}^{-1}A_{21}\|_\infty & 0 & \|A_{22}^{-1}A_{23}\|_\infty & \cdots & \|A_{22}^{-1}A_{2p}\|_\infty \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \|A_{pp}^{-1}A_{p1}\|_\infty & \cdots & \cdots & \|A_{pp}^{-1}A_{p,p-1}\|_\infty & 0 \end{bmatrix}$$

we can formulate a sufficient condition on the convergence of the forward Monte Carlo scheme:

$$\|\tilde{H}\|_\infty < 1 \Rightarrow \|\hat{H}\|_\infty < 1. \quad (22)$$

Note that (21) also implies that  $\|H\|_\infty < 1$ .

We now turn our attention to the adjoint method. Analogously to the forward method, we can define

$$(\hat{H})_{ij}^T = |H_{ij}^T| \left( \sum_{k=1}^n |H_{ik}^T| \right) = |H_{ji}| \left( \sum_{k=1}^n |H_{ki}| \right).$$

This allows us to formulate a sufficient condition for the convergence of the adjoint Monte Carlo method with block diagonal preconditioning. Letting

$$\tilde{H} = \begin{bmatrix} 0 & \|A_{11}^{-1}A_{12}\|_1 & \cdots & \cdots & \|A_{11}^{-1}A_{1p}\|_1 \\ \|A_{22}^{-1}A_{21}\|_1 & 0 & \|A_{22}^{-1}A_{23}\|_1 & \cdots & \|A_{22}^{-1}A_{2p}\|_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \|A_{pp}^{-1}A_{p1}\|_1 & \cdots & \cdots & \|A_{pp}^{-1}A_{p,p-1}\|_1 & 0 \end{bmatrix},$$

a sufficient condition is that  $\|\tilde{H}\|_1 < 1$ , i.e.,

$$\sum_{\substack{h=1 \\ h \neq k}}^p \|A_{hh}^{-1}A_{hk}\|_1 < 1, \quad \forall k = 1, \dots, p. \quad (23)$$

Again, this condition also implies that  $\|H\|_1 < 1$ .

We say that  $A$  is *strictly block diagonally dominant by rows (columns) with respect to a given block partition* if condition (21) (respectively, (23)) is satisfied relative to that particular block partition. Note that a matrix may be strictly block diagonally dominant with respect to one partition and not to another. We note that these definitions of block diagonal dominance are different from those found in, e.g., [15], and they are easier to check in practice since they do not require computing the 2-norm of the blocks of  $H$ .

### 3.4. Adaptive methods

In formulas (4) and (10), the estimation of the solution to the linear system (1) involves infinite sums, which in actual computation have to be truncated. In the following we discuss criteria to decide the number of steps to be taken in a single random walk as well as the number of random walks that need to be performed at each Richardson iteration.

**3.4.1. History length** We first consider criteria to terminate an individual random walk, effectively deciding how many terms of the Neumann series will be considered. One possibility is to set a predetermined history length, at which point all histories are terminated. This approach, however, presents two difficulties. First, it is difficult to determine a priori how many steps on average will be necessary to achieve a specified tolerance. Second, due to the stochastic nature of the random walks, some histories will retain important information longer than others. Truncating histories at a predetermined step runs the risk of either prematurely truncating important histories, leading to larger errors, or continuing unimportant histories longer than necessary, leading to computational inefficiency.

Our goal is to apply a cutoff via an automatic procedure, without requiring any user intervention. We would like to determine an integer  $m$  such that

$$\begin{aligned} \tilde{\theta} &= E \left[ \sum_{\ell=0}^m W_\ell f_{k_\ell} \right] \\ &= f_i + \sum_{\ell=1}^m \sum_{k_1=1}^n \sum_{k_2=1}^n \cdots \sum_{k_\ell=1}^n P_{i,k_1} P_{k_1,k_2} \cdots P_{k_{\ell-1},k_\ell} w_{k_0,k_1} w_{k_1,k_2} \cdots w_{k_{\ell-1},k_\ell} f_{k_\ell} \end{aligned}$$

and

$$\begin{aligned} \tilde{\theta}_i &= E \left[ \sum_{\ell=0}^m W_\ell \delta_{k_\ell, j} \right] \\ &= \sum_{\ell=0}^m \sum_{k_1}^n \sum_{k_2}^n \cdots \sum_{k_\ell}^n f_{k_0} P_{k_0, k_1} P_{k_1, k_2} \cdots P_{k_{\ell-1}, k_\ell} w_{k_0, k_1} \cdots w_{k_{\ell-1}, k_\ell} \delta_{k_\ell, i} \end{aligned}$$

are good approximations of (4) and (10), respectively.

In [26] a criterion was given which is applicable to both forward and adjoint methods. It requires to set up a relative weight cutoff threshold  $W_c$  and to look for a step  $m$  such that

$$W_m \leq W_c W_0. \quad (24)$$

In (24),  $W_0$  is the value of the weight at the initial step of the random walk and  $W_m$  is the value of the weight after  $m$  steps. We will adopt a similar strategy in this work.

**3.4.2. Number of random walks** We now consider the selection of the number of random walks that should be performed to achieve a given accuracy. Unlike the termination of histories, this is a subject that has not been discussed in the Monte Carlo linear solver literature, as all previous studies have considered the simulation of a prescribed number of histories.

The expression for the variance of the forward method is given by formula (5). In this context, a reasonable criterion to determine the number  $\tilde{N}_i$  of random walks to be run is to set a threshold  $\varepsilon_1$  and determine  $\tilde{N}_i$  such that

$$\frac{\sqrt{Var[\theta_i]}}{|E[\theta_i]|} < \varepsilon_1, \quad i = 1, \dots, n. \quad (25)$$

The dependence of  $Var[\theta_i]$  and  $E[\theta_i]$  on  $\tilde{N}_i$  is due to the fact that  $\theta_i$  is estimated by performing a fixed number of histories. Therefore, we are controlling the relative standard deviation, requiring it not to be too large. In other words, we are aiming for an approximate solution in which the uncertainty factor is small relative to the expected value. This simple adaptive approach can be applied for the estimation of each component  $x_i$ . Hence, a different number of histories may be employed to compute different entries of the solution vector.

As concerns the adjoint method, the estimation of the variance is given in formula (11). A possible criterion for the adaptive selection of the number  $\tilde{N}$  of random walk, in this situation, is that it satisfies the condition

$$\frac{\|\sigma^{\tilde{N}}\|_1}{\|\mathbf{x}\|_1} < \varepsilon_1, \quad (26)$$

where  $\sigma$  is a vector whose entries are  $\sigma_i^{\tilde{N}} = Var[\theta_i]$  and  $\mathbf{x}$  is a vector whose entries are  $x_i = E[\theta_i]$ .

The criteria introduced above can be exploited to build an a posteriori adaptive algorithm, capable of identifying the minimal value of  $\tilde{N}$  that verifies (25) or (26), respectively. Algorithms 4 and 5 describe the Monte Carlo approaches with the adaptive criteria.

---

**Algorithm 4:** A posteriori adaptive Forward Monte Carlo

---

**Data:**  $N, \varepsilon_1$   
**Result:**  $\tilde{N}_i, \sigma_i, x_i$

```

1 for  $i = 1, \dots, n$  do
2    $\tilde{N}_i = N$ ;
3   compute  $x_i$ ;
4   compute  $\sigma_i$ ;
5   while  $\frac{\sigma_i}{|x_i|} < \varepsilon_1$  do
6      $\tilde{N} = \tilde{N} + N$ ;
7     compute  $x_i$ ;
8     compute  $\sigma_i$ ;
9   end
10 end
```

---



---

**Algorithm 5:** A posteriori adaptive Adjoint Monte Carlo

---

**Data:**  $N, \varepsilon_1$   
**Result:**  $\tilde{N}, \sigma, \mathbf{x}$

```

1  $\tilde{N} = N$ ;
2 compute  $\mathbf{x}$ ;
3 compute  $\sigma$ ;
4 while  $\frac{\|\sigma\|_1}{\|\mathbf{x}\|_1} < \varepsilon_1$  do
5    $\tilde{N} = \tilde{N} + N$ ;
6   compute  $\mathbf{x}$ ;
7   compute  $\sigma$ ;
8 end
```

---

The use of the adaptive approach for the selection of the number of histories has a dual purpose. First, it guarantees that the update computed with the Monte Carlo step is accurate enough to preserve convergence. Second, it provides the user with a tuning parameter to distribute the computation between the deterministic and the stochastic part of the algorithm. Lowering the value of the threshold for the relative standard deviation increases the number of histories per iteration. This results in a more accurate stochastic updating and reduces the iterations necessary to converge. While guessing an a priori fixed number of histories may lead to a smaller number of Monte Carlo histories overall, it might either hinder the convergence or distribute too much computation on the deterministic side of the scheme (or both). Generally speaking, the adaptive approach is more robust and more useful, especially when  $\rho(H)$  is close to 1, since in this case the Richardson step is less effective in dampening the uncertainty coming from the previous iterations.

### 3.5. Preconditioning

As noted at the end of Section 2, left preconditioning can be incorporated into any Monte Carlo linear solver algorithm by simply substituting  $A$  with  $P^{-1}A$  and  $\mathbf{b}$  with  $P^{-1}\mathbf{b}$  in (1); i.e., we set  $H = I - P^{-1}A$  and  $\mathbf{f} = P^{-1}\mathbf{b}$  in (2). Right preconditioning can also be incorporated by rewriting (1) as  $AP^{-1}\mathbf{y} = \mathbf{b}$ , with  $\mathbf{y} = P\mathbf{x}$ ; i.e., we set  $H = I - AP^{-1}$  and replace  $\mathbf{x}$  by  $\mathbf{y}$  in (2). The solution  $\mathbf{x}$  to the original system (1) is then given by  $\mathbf{x} = P^{-1}\mathbf{y}$ . Likewise, split (left and right) preconditioning can also be used. The Monte Carlo process, however, imposes some constraints on the choice of preconditioner. Most significantly, because the transition probabilities are built based on the values of the iteration matrix  $H$ , it is necessary to have access to the entries of the preconditioned matrix  $P^{-1}A$  (or  $AP^{-1}$ ). Therefore, we are limited to preconditioners that enable

explicitly forming the preconditioned matrix while retaining some of the sparsity of the original matrix.

One possible preconditioning approach involves either diagonal or block diagonal preconditioning (with blocks of small or moderate size). Diagonal preconditioning does not alter the sparsity of the original coefficient matrix, while block diagonal preconditioning will incur a moderate amount of fill-in in the preconditioned matrix if the blocks are not too large. From the discussions in Section 3.3.1 and 3.3.3, selecting a diagonal or block diagonal preconditioner guarantees convergence of the Monte Carlo schemes for matrices that are strictly diagonally dominant or block diagonally dominant, respectively. In addition,  $M$ -matrices that are not strictly or block diagonally dominant can also be dealt with by first rescaling  $A$  so that it becomes strictly diagonally dominant, as discussed in Section 3.3.1.

In principle, other standard preconditioning approaches can also be used in an attempt to achieve both  $\rho(H) < 1$  and  $\rho(\hat{H}) < 1$  while still retaining sparsity in the preconditioned matrix. One possibility is the use of incomplete LU factorizations [4, 24]. If  $P = LU$  is the preconditioner with sparse triangular factors  $L$  and  $U$ , then  $P^{-1}A$  can in principle be formed explicitly provided that the sparsity in  $L$ ,  $U$  and  $A$  is carefully exploited in the forward and back substitutions needed to form  $(LU)^{-1} = U^{-1}(L^{-1}A)$ . In general, however, this results in a rather full preconditioned matrix; sparsity needs to be preserved by dropping small entries in the resulting matrix.

Another class of preconditioners that are potentially of interest for use with Monte Carlo linear solvers are approximate inverse preconditioners [4]. In these algorithms, an approximation to the inverse of  $A$  is generated directly and the computation of the preconditioned matrix reduces to one or more sparse matrix-matrix products, a relatively straightforward task. As with ILU factorizations, multiple versions of approximate inverse preconditioning exist which may have different behavior in terms of effectiveness of the preconditioner versus the resulting reduction in sparsity.

A downside of the use of ILU or approximate inverse preconditioning is that the quality of preconditioner needed to achieve both  $\rho(H) < 1$  and  $\rho(\hat{H}) < 1$  is difficult to determine. Indeed, in some situations it may happen that modifying the preconditioner so as to reduce  $\rho(H)$  may actually lead to an increase in  $\rho(\hat{H})$ , decreasing the effectiveness of the Monte Carlo process on the system or even causing it to diverge. In other words, for both ILU and sparse approximate inverse preconditioning it seems to be very difficult to guarantee convergence of the Monte Carlo linear solvers a priori.

### 3.6. Considerations about computational complexity

Providing an analysis of the computational complexity for the aforementioned algorithms is not entirely straightforward because of their stochastic nature. Indeed, different statistical samplings can produce estimates with different uncertainty levels, requiring a proper tuning of the number of samplings computed to reach a prescribed accuracy. Moreover, we already mentioned that the asymptotic analysis of MC convergence assumes random walks with infinitely many steps and  $N \rightarrow \infty$ , where  $N$  is the number of random walks. However, in practice each history must be truncated to a finite number of steps and the number of statistical samplings must be finite as well. The actual value of  $N$  and the length of the histories affect the accuracy of the statistical estimation, thus influencing the number of iterations in a hybrid algorithm, since the uncertainty propagates to subsequent iterations. Therefore, here we can only provide a tentative analysis of the complexity of the forward and adjoint Monte Carlo methods, assuming a specific history length and a fixed number  $N$  of statistical samplings.

Recalling formula (4) for the entry-wise estimate for the solution with the forward method, the cost of reconstructing the entire solution vector is

$$\text{Forward: } \mathcal{O}(N \cdot k_\ell \cdot n), \quad (27)$$

where  $N$  is the number of histories,  $k_\ell$  is the length of each random walk, and  $n$  is the number of unknowns. As concerns the adjoint method, formula (10) leads to

$$\text{Adjoint: } \mathcal{O}(N \cdot k_\ell). \quad (28)$$

The operation count for the hybrid schemes can thus be obtained by combining the cost of the Richardson scheme with the complexity of standard MC techniques.

Regarding the Sequential Monte Carlo algorithm, the standard MC scheme is combined with the computation of the residual at each iteration. The cost of the residual computation is essentially that of a sparse matrix-vector product, which is  $\mathcal{O}(n)$  for a sparse system. Therefore, the complexity of a single Sequential MC iteration is

$$\text{Sequential MC with forward MC update: } \mathcal{O}((n+1) \cdot N \cdot k_\ell) \quad (29)$$

using the forward MC as an inner solver, and

$$\text{Sequential MC with adjoint MC update: } \mathcal{O}(n + N \cdot k_\ell) \quad (30)$$

using the adjoint MC as an inner solver.

A single iteration of MCSA requires computing the residual, applying a matrix-vector product using  $H$ , and applying an MC update. The complexity of an MCSA iteration using the forward MC is therefore

$$\text{MCSA with forward MC update: } \mathcal{O}((2n+1) \cdot N \cdot k_\ell), \quad (31)$$

whereas using the adjoint MC we find

$$\text{MCSA with adjoint MC update: } \mathcal{O}(2n + N \cdot k_\ell). \quad (32)$$

Note that these estimates require  $\text{nnz}(H) \approx \text{nnz}(A)$ , in the sense that both  $H$  and  $A$  contain  $\mathcal{O}(n)$  nonzeros. The actual values attained by  $N$  and  $k_\ell$  depend on the thresholds employed to truncate a single history and to determine the number of random walks to use. In general, the higher  $N$  and  $k_\ell$ , the lower the number of outer iterations to achieve a prescribed accuracy. Finally, the total cost will depend also on the number of iterations, which is difficult to predict in practice.

## 4. NUMERICAL RESULTS

In this section we discuss the results of numerical experiments. The main goal of these experiments is to gain some insight into the behavior of adaptive techniques, and to test different preconditioning options within the Monte Carlo approach.

### 4.1. Numerical tests with adaptive methods

In this subsection we study experimentally the adaptive approaches discussed in Section 3.4. For this purpose, we restrict our attention to standard Monte Carlo linear solvers.

For these tests we limit ourselves to small matrices, primarily because the numerical experiments are being computed on a standard laptop and the computational cost of Monte Carlo methods rapidly becomes prohibitive on such machines. The smallest of these matrices represents a finite difference discretization of a 1D reaction-diffusion equation, the second one is a discrete 2D Laplacian with zero Dirichlet boundary conditions, the third a steady 2D advection-diffusion operator discretized by quadrilateral linear finite elements using the IFISS package [12], and the fourth one results from a finite volume discretization of a thermal radiation diffusion equation (Marshak problem). The first and the last of these matrices are strictly diagonally dominant by both rows and columns. For all the problems, left diagonal preconditioning is applied. Details about these matrices are given in Table I.

We present results for both the Forward and the Adjoint Monte Carlo methods. As concerns the Forward method, we set the maximum number of histories per entry to  $10^7$ . In Tables II and III we show results for values of the adaptive threshold (25) equal to  $\varepsilon_1 = 0.1$  and  $\varepsilon_1 = 0.01$ , respectively. A batch size of two is used at each adaptive check to verify the magnitude of the apparent relative standard deviation. As expected, results are aligned with the convergence rate predicted by the Central Limit Theorem. Indeed, decreasing by a factor of 10 the tolerance  $\varepsilon_1$  we see that the relative error undergoes a decrease of the same order, requiring roughly one hundred times more histories.

Type of problem	$n$	$\rho(H)$	Forward $\rho(\hat{H})$	Adjoint $\rho(\hat{H})$
1D reaction-diffusion	50	0.4991	0.9827	0.9827
2D Laplacian	900	0.9949	0.994	0.9945
2D advection-diffusion	1089	0.9836	0.983	0.983
Marshak problem	1600	0.6009	0.3758	0.3815

Table I. Properties of the matrices employed for the checks on adaptivity.

Type of problem	Relative Error	Nb. Histories
1D reaction-diffusion	0.003	5, 220
2D Laplacian	0.1051	262, 350
2D advection-diffusion	0.022	1, 060, 770
Marshak problem	0.0012	1, 456, 000

Table II. Forward Monte Carlo. Adaptive selection of histories,  $\varepsilon_1 = 0.1$ .

Type of problem	Relative Error	Nb. Histories
1D reaction-diffusion	$4 \cdot 10^{-4}$	512, 094
2D Laplacian	0.0032	108, 551, 850
2D advection-diffusion	0.0023	105, 476, 650
Marshak problem	$5.8 \cdot 10^{-4}$	144, 018, 700

Table III. Forward Monte Carlo. Adaptive selection of histories,  $\varepsilon_1 = 0.01$ .

Type of problem	Relative Error	Nb. Histories
1D reaction-diffusion	0.08	1820
2D Laplacian	0.136	570
2D advection-diffusion	0.08	2400
Marshak problem	0.288	880

Table IV. Collision estimator – Adjoint Monte Carlo. Adaptive selection of histories,  $\varepsilon_1 = 0.1$ .

Type of problem	Relative Error	Nb. Histories
1D reaction-diffusion	0.0082	185, 400
2D Laplacian	0.0122	126, 800
2D advection-diffusion	0.0093	219, 293
Marshak problem	0.0105	650, 040

Table V. Collision estimator – Adjoint Monte Carlo. Adaptive selection of histories,  $\varepsilon_1 = 0.01$ .

In the case of the 2D Laplacian we actually have an increase of a factor close to 400 in the number of histories, but the relative error is decreased by more than thirty times. For this particular example, the forward method overestimates the number of histories needed to satisfy a prescribed reduction on the standard deviation.

As regards the Adjoint Monte Carlo, at each adaptive check the number of random walks employed is increased by ten. A maximum number of histories equal to  $10^{10}$  is set. Tables IV, V and VI show results for the different test cases using adaptive thresholds (26) with values  $\varepsilon_1 = 0.1$ ,  $\varepsilon_1 = 0.01$  and  $\varepsilon_1 = 0.001$ , respectively. By comparing the reported errors, it is clear that a decrease in the value of the threshold induces a reduction of the relative error of the same order of magnitude. This confirms the effectiveness of the adaptive selection of histories with an error reduction goal. Each decrease in the error by an order of magnitude requires an increase in the total number of histories employed by two orders of magnitude, as expected.

Type of problem	Relative Error	Nb. Histories
1D reaction-diffusion	$9.56 \cdot 10^{-4}$	15,268,560
2D Laplacian	0.001	12,600,000
2D advection-diffusion	0.0011	23,952,000
Marshak problem	0.0011	80,236,000

Table VI. Collision estimator – Adjoint Monte Carlo. Adaptive selection of histories,  $\varepsilon_1 = 0.001$ .

Type of problem	Relative Error	Nb. Histories
1D reaction-diffusion	0.0463	1000
2D Laplacian	0.1004	900
2D advection-diffusion	0.0661	1300
Marshak problem	0.0526	2000

Table VII. Expected value estimator – Adjoint Monte Carlo. Adaptive selection of histories,  $\varepsilon_1 = 0.1$ .

Type of problem	Relative Error	Nb. Histories
1D reaction-diffusion	0.004	100,600
2D Laplacian	0.0094	83,700
2D advection-diffusion	0.0088	124,400
Marshak problem	0.0056	166,000

Table VIII. Expected value estimator – Adjoint Monte Carlo. Adaptive selection of histories,  $\varepsilon_1 = 0.01$ .

Type of problem	Relative Error	Nb. Histories
1D reaction-diffusion	0.004	10,063,300
2D Laplacian	$9.31 \cdot 10^{-4}$	8,377,500
2D advection-diffusion	0.0013	12,435,000
Marshak problem	$7.79 \cdot 10^{-4}$	16,537,000

Table IX. Expected value estimator – Adjoint Monte Carlo. Adaptive selection of histories,  $\varepsilon_1 = 0.001$ .

The same simulations can be run resorting to the expected value estimator. Results are shown in the Tables VII, VIII and IX for the threshold values of  $\varepsilon_1 = 0.1$ ,  $\varepsilon_1 = 0.01$  and  $\varepsilon_1 = 0.001$  respectively. As it can be noticed, in terms of error scaling the results are quite similar to the ones obtained with the collision estimator. As regards the number of histories needed to reach a prescribed accuracy, the orders of magnitude are the same for both the collision and the expected value estimators. However, the expected value estimator requires in most cases a smaller number of realizations. This behavior becomes more pronounced as the value of the threshold decreases, making the computation increasingly cost-effective.

#### 4.2. Preconditioning approaches

In this subsection we examine the effect of different preconditioners on the values attained by the spectral radii  $\rho(H)$  and  $\rho(\hat{H})$ . For this purpose, we focus on the 2D discrete Laplacian and the 2D discrete advection diffusion operator from the previous section.

The values of the two spectral radii with diagonal preconditioning have already been shown in Table IV. Here we consider the effect of block diagonal preconditioning for different block sizes, and the use of the factorized sparse approximate inverse preconditioner AINV [5, 6] for different values of the drop tolerance (which controls the sparsity in the approximate inverse factors). Intuitively, with these two types of preconditioners both  $\rho(H)$  and  $\rho(\hat{H})$  should approach zero for increasing block size and decreasing drop tolerance, respectively; however, the convergence need not be monotonic in general, particularly for  $\rho(\hat{H})$ . This somewhat counterintuitive behavior is shown in Tables X and XI, where an increase in the size of the blocks used for the block diagonal

Block size	$\frac{nnz(H)}{nnz(A)}$	$\rho(H)$	$\rho(\hat{H})$
5	2.6164	0.9915	0.9837
10	5.6027	0.9907	0.9861
30	16.3356	0.9898	0.9886

Table X. Behavior of  $\rho(H)$  and  $\rho(\hat{H})$  for the 2D Laplacian with block diagonal preconditioning.

Block size	$\frac{nnz(H)}{nnz(A)}$	$\rho(H)$	$\rho(\hat{H})$
3	1.4352	0.9804	0.9531
9	3.0220	0.9790	0.9674
33	9.8164	0.9783	0.9774

Table XI. Behavior of  $\rho(H)$  and  $\rho(\hat{H})$  for the 2D advection-diffusion problem with block diagonal preconditioning.

Drop tolerance	$\frac{nnz(H)}{nnz(A)}$	$\rho(H)$	$\rho(\hat{H})$
0.05	8.2797	0.9610	1.0231
0.01	33.1390	0.8668	0.8279

Table XII. Behavior of  $\rho(H)$  and  $\rho(\hat{H})$  for the 2D Laplacian with AINV preconditioning.

Drop tolerance	$\frac{nnz(H)}{nnz(A)}$	$\rho(H)$	$\rho(\hat{H})$
0.05	3.8392	0.9396	0.9069
0.01	15.1798	0.7964	0.6361

Table XIII. Behavior of  $\rho(H)$  and  $\rho(\hat{H})$  for the 2D advection-diffusion problem with AINV preconditioning.

preconditioner results in an increase of  $\rho(\hat{H})$  for both test problems. Note also the very slow rate of decrease in  $\rho(H)$  for increasing block size, which is more than offset by the rapid increase in the density of  $H$ , which of course implies much higher costs. We mention that for a block size of 30 the 2D discrete Laplacian is block diagonally dominant, but not for smaller block sizes. The 2D discrete advection-diffusion operator is not block diagonally dominant for any of the three reported block sizes.

In Tables XII and XIII the values of the spectral radii are shown for the AINV preconditioner with two different values of the drop tolerance [5, 6]. It is interesting to point out that, for the two-dimensional Laplacian, a drop tolerance  $\tau = 0.05$  entails  $\rho(H) < 1$  but the same does not hold for  $\hat{H}$ . Both convergence conditions are satisfied by reducing the drop tolerance, but at the price of very high fill-in in the preconditioned matrix.

In summary, we conclude that it is generally very challenging to guarantee the convergence of Monte Carlo linear solvers a priori. Simple (block) diagonal preconditioners may work even if  $A$  is not strictly (block) diagonally dominant, but it is hard to know beforehand if a method will converge, especially due to lack of a priori bounds on  $\rho(\hat{H})$ . Moreover, the choice of the block sizes in the block diagonal case is not an easy matter. Sparse approximate inverses are a possibility but the amount of fill-in required to satisfy the convergence conditions could be unacceptably high. These observations suggest that it is difficult to use Monte Carlo linear solvers in the case of linear systems arising from the discretization of steady-state PDEs, which typically are not strictly diagonally dominant. As we shall see later, the situation is more favorable in the case of time-dependent problems.

### 4.3. Hybrid methods

Next, we present results for hybrid methods, combining a deterministic Richardson iteration with Monte Carlo acceleration. We recall that the convergence conditions are the same as for the direct stochastic approaches, therefore the concluding observations from the previous section still apply.

4.3.1. *Poisson problem* Consider the standard 2D Poisson model problem:

$$\begin{cases} -\Delta u = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases} \quad (33)$$

where  $\Omega = (0, 1) \times (0, 1)$ . For the numerical experiments we use as the right-hand side a sinusoidal distribution in both  $x$  and  $y$  directions:

$$f(x, y) = \sin(\pi x) \sin(\pi y).$$

We discretize problem (33) using standard 5-point finite differences. For  $N = 32$  nodes in each direction, we obtain the  $900 \times 900$  linear system already used in the previous section.

Consider the iteration matrix  $H$  corresponding to (left) diagonal preconditioning. It is well known that  $\rho(H) < 1$ , and indeed we can see from Table I that  $\rho(H) \approx 0.9949$ . It is also easy to see that  $\|H\|_1 = 1$ . In order for the Adjoint Monte Carlo method to converge, it is necessary to have  $\rho(\hat{H}) < 1$ , too. If an almost optimal probability is used, the Adjoint Monte Carlo method leads to a  $\hat{H}$  matrix such that

$$\|\hat{H}\|_1 \leq \|H\|_1^2 = 1.$$

This condition by itself is not enough to guarantee that  $\rho(\hat{H}) < 1$ . However, the iteration matrix  $H$  has zero-valued entries on the main diagonal and it has:

- four entries equal to  $\frac{1}{4}$  on the rows associated with a node which is not adjacent to the boundary;
- two entries equal to  $\frac{1}{4}$  on the rows associated with nodes adjacent to the corner of the square domain;
- three entries equal to  $\frac{1}{4}$  on the rows associated with nodes adjacent to the boundary on an edge of the square domain.

Recalling the definition of  $\hat{H}$  in terms of the entries of the iteration matrix  $H$  and the transition probability  $P$ , we see that

$$\hat{H} = \tilde{D}H,$$

where  $\tilde{D}$  is a diagonal matrix with diagonal entries equal to 1,  $\frac{1}{2}$  or  $\frac{3}{4}$ . In particular,  $\tilde{d}_i = \text{diag}(\tilde{D})_i = 1$  if the  $i$ th row of the discretized Laplacian is associated with a node which is not adjacent to the boundary,  $\tilde{d}_i = \text{diag}(\tilde{D})_i = \frac{1}{2}$  if the row is associated with a node of the grid adjacent to the corner of the boundary, and  $\tilde{d}_i = \text{diag}(\tilde{D})_i = \frac{3}{4}$  if the associated node of the grid is adjacent to the boundary of the domain far from a corner. Since  $H$  is an irreducible nonnegative matrix and  $\tilde{D}$  is a positive definite diagonal matrix, we can invoke a result in [17] to conclude that

$$\rho(\hat{H}) = \rho(\tilde{D}H) \leq \rho(\tilde{D})\rho(H).$$

Since  $\rho(\tilde{D}) = 1$ , then  $\rho(\hat{H}) \leq \rho(H) < 1$ . Therefore, diagonal preconditioning always guarantees convergence for the Monte Carlo linear solver applied to any 2D Laplace operator discretized with 5-point finite differences. Similar arguments apply to the  $d$ -dimensional Laplacian, for any  $d$ .

Results of numerical experiments are reported in Table XIV. We compare the purely deterministic Richardson iteration (Jacobi's method in this case) with two hybrid methods, Halton's sequential Monte Carlo and MCSA using the Adjoint method. The threshold for the adaptive selection of the random walks is set to  $\varepsilon_1 = 0.1$ . Convergence is attained when the initial residual has been reduced by at least eight orders of magnitude, starting from a zero initial guess. Note that, as expected, the

algorithm	relative err.	# iterations	average # histories per iteration
Richardson	$9.8081 \cdot 10^{-8}$	3133	-
Sequential MC	$7.9037 \cdot 10^{-8}$	9	8,264,900
Adjoint MCSA	$8.0872 \cdot 10^{-8}$	8	1,738,250

Table XIV. Numerical results for the Poisson problem.

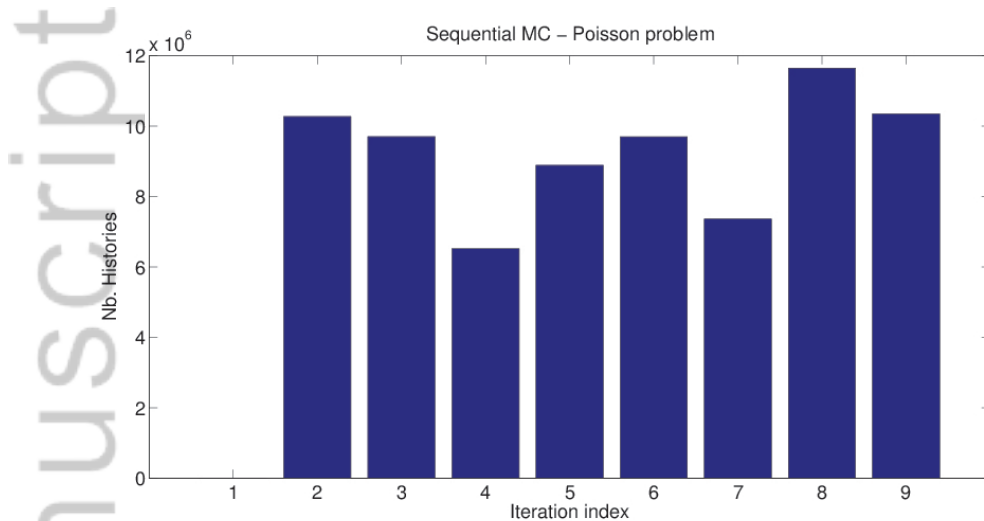


Figure 1. Sequential MC - Poisson problem. Number of random walks employed at each iteration. The number for the first iteration is 2000.

diagonally preconditioned Richardson iteration converges very slowly, since the spectral radius of  $H$  is very close to 1. Halton’s Sequential Monte Carlo performs quite well on this problem, but even better results are obtained by the Adjoint MCSA method, which converges in one iteration less than Sequential Monte Carlo while requiring far less Monte Carlo histories per iteration. This might be explained by the fact that within each outer iteration, the first relaxation step in the MCSA algorithm refines the accuracy of the solution coming from the previous iteration, before using it as the input for the MC solver at the current iterate. In particular, its effect is to dampen the statistical noise produced by the Monte Carlo linear solver in the estimation of the update  $\delta x^{l+\frac{1}{2}}$  performed at the previous iteration. Therefore, the refinement, or smoothing, accomplished by the Richardson relaxation decreases the number of random walks needed for a prescribed accuracy. This hypothesis is validated by the fact that at the first iteration both Sequential Monte Carlo and MCSA use the same number of histories. Their behaviors start differing from the second iteration on, when the statistical noise is introduced in the estimation of the correction to the current iterate; see Figures 1 and 2. It can be seen that in all the numerical tests presented in this section there is a sharp increase (from  $O(10^3)$ – $O(10^5)$  to  $O(10^6)$ – $O(10^7)$ ) in the number of histories when going from the first to subsequent iterations. This is due to the introduction of statistical noise coming from the Monte Carlo updating of the solution. Indeed, the stochastic noise, introduced from the second iteration on, increases the uncertainty associated with the estimate of the solution. Therefore, the adaptive criterion forces the algorithm to perform a higher number of histories to achieve a prescribed accuracy.

4.3.2. *Reaction-diffusion problem* Here we consider the simple reaction-diffusion problem

$$\begin{cases} -\Delta u + \sigma u = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases} \quad (34)$$

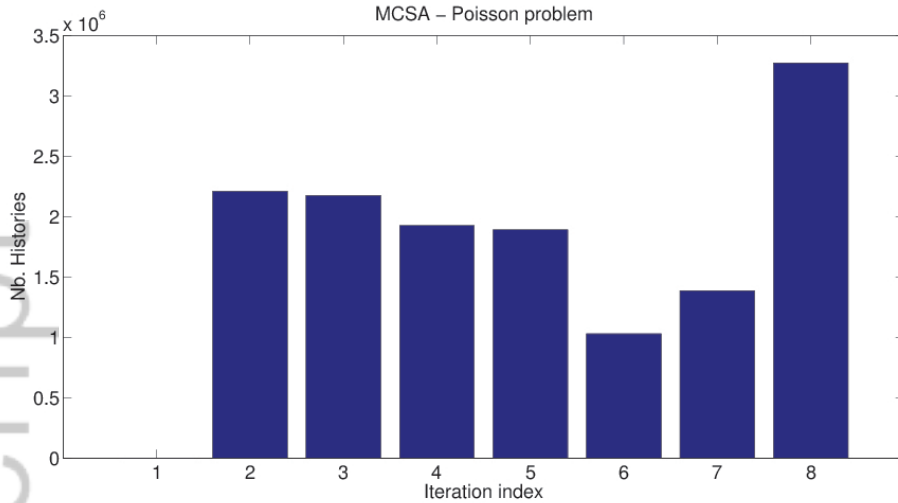


Figure 2. MCSA - Poisson problem. Number of random walks employed at each iteration. The number for the first iteration is 2000.

algorithm	relative err.	# iterations	average # histories per iteration
Richardson	$9.073 \cdot 10^{-8}$	634	-
Sequential MC	$8.415 \cdot 10^{-8}$	8	12,391,375
Adjoint MCSA	$6.633 \cdot 10^{-8}$	7	3,163,700

Table XV. Numerical results for the diffusion reaction problem.

where  $\Omega = (0, 1) \times (0, 1)$ ,  $\sigma = 0.1$  and  $f \equiv 1$ . A 5-point finite difference scheme is applied to discretize the problem. The number of nodes on each direction of the domain is 100, so that  $h \approx 0.01$ . The discretized problem is  $n \times n$  with  $n = 9604$ . A left diagonal preconditioning is again applied to the coefficient matrix obtained from the discretization, which is strictly diagonally dominant. The 1-norm of the iteration matrix is  $\|H\|_1 \approx 0.9756$ . This automatically guarantees the convergence of the Adjoint Monte Carlo linear solver. In Table XV a comparison between the deterministic Richardson iteration, Sequential Monte Carlo and MCSA is provided. The results are similar to those for the Poisson equation. The number of histories per iteration for Sequential Monte Carlo and MCSA is shown in Figures 3 and 4.

4.3.3. *Parabolic problem* Here we consider the following time-dependent problem:

$$\begin{cases} \frac{\partial u}{\partial t} - \mu \Delta u + \beta(\mathbf{x}) \cdot \nabla u = 0, & \mathbf{x} \in \Omega, \quad t \in (0, T] \\ u(\mathbf{x}, 0) = u_0, & \mathbf{x} \in \Omega \\ u(\mathbf{x}, t) = u_D(\mathbf{x}), & \mathbf{x} \in \partial\Omega, \quad t \in (0, T], \end{cases} \quad (35)$$

where  $\Omega = (0, 1) \times (0, 1)$ ,  $T > 0$ ,  $\mu = \frac{3}{200}$ ,  $\beta(\mathbf{x}) = [2y(1-x^2), -2x(1-y^2)]^T$ , and  $u_D = 0$  on  $\{\{x=0\} \times (0, 1)\}, \{(0, 1) \times \{y=0\}\}, \{(0, 1) \times \{y=1\}\}$ .

Implicit discretization in time (backward Euler scheme) with time step  $\Delta t$  and a spatial discretization with quadrilateral linear finite elements using the IFISS toolbox [12] leads to a sequence of linear systems of the form

$$\left( \frac{1}{\Delta t} M + A \right) \mathbf{u}^{k+1} = \mathbf{f}^k, \quad k = 0, 1, \dots$$

Here we restrict our attention to a single generic time step. The right-hand side is chosen so that the exact solution to the linear system for the specific time step chosen is the vector of all ones. For

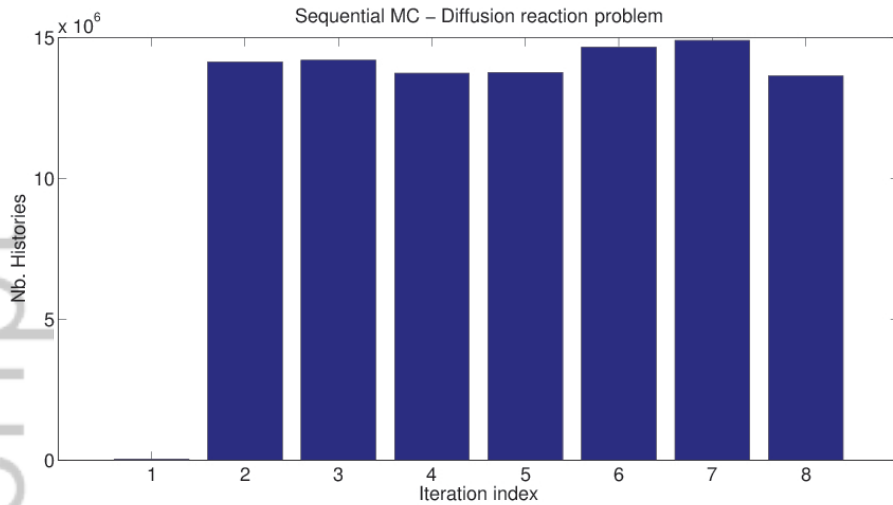


Figure 3. Sequential MC - Reaction-diffusion problem. Number of random walks employed at each iteration. The number first the first iteration is 36,000.

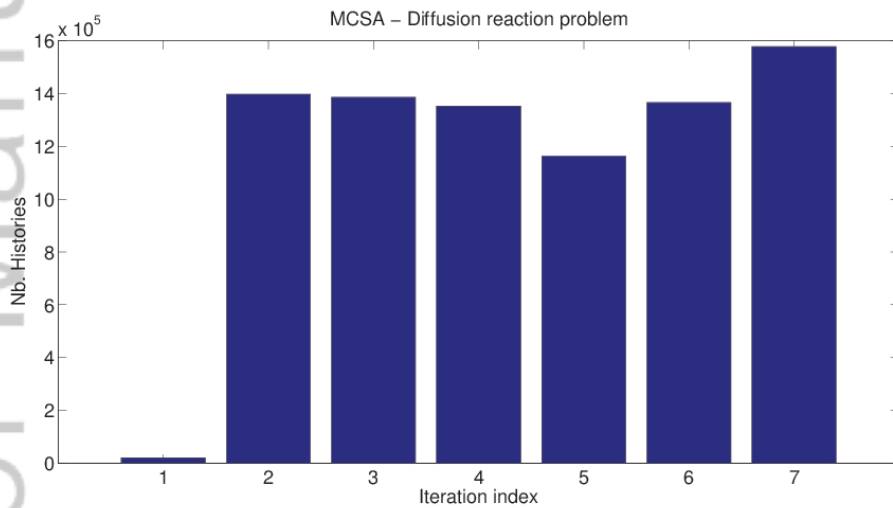


Figure 4. MCSA - Reaction-diffusion problem. Number of random walks employed at each iteration. The number first the first iteration is 36,000.

the experiments we use a uniform discretization with mesh size  $h = 2^{-8}$  and we let  $\Delta t = 10h$ . The resulting linear system has  $n = 66,049$  unknowns.

We use the factorized sparse approximate inverse AINV [6] as a right preconditioner, with drop tolerance  $\tau = 0.05$  for both inverse factors. With this choice, the spectral radius of the iteration matrix  $H = I - AP^{-1}$  is  $\rho(H) \approx 0.9218$  and the spectral radius of  $\hat{H}$  for the Adjoint Monte Carlo is  $\rho(\hat{H}) \approx 0.9148$ . The MAO transition probability is employed. Resorting to a uniform probability in this case would have impeded the convergence, since in this case  $\rho(\hat{H}) \approx 1.8401$ . This is an example demonstrating how the MAO probability can improve the behavior of the stochastic algorithm, outperforming the uniform one.

The fill-in ratio is given by  $\frac{nnz(H)}{nnz(A)} = 4.26$ , therefore the relative number of nonzero elements in  $H$  is still acceptable in terms of storage and computational costs.

algorithm	relative err.	# iterations	average # histories per iteration
Richardson	$6.277 \cdot 10^{-7}$	178	-
Sequential MC	$1.918 \cdot 10^{-9}$	8	51,535,375
Adjoint MCSA	$1.341 \cdot 10^{-9}$	6	16,244,000

Table XVI. Numerical results for the parabolic problem.

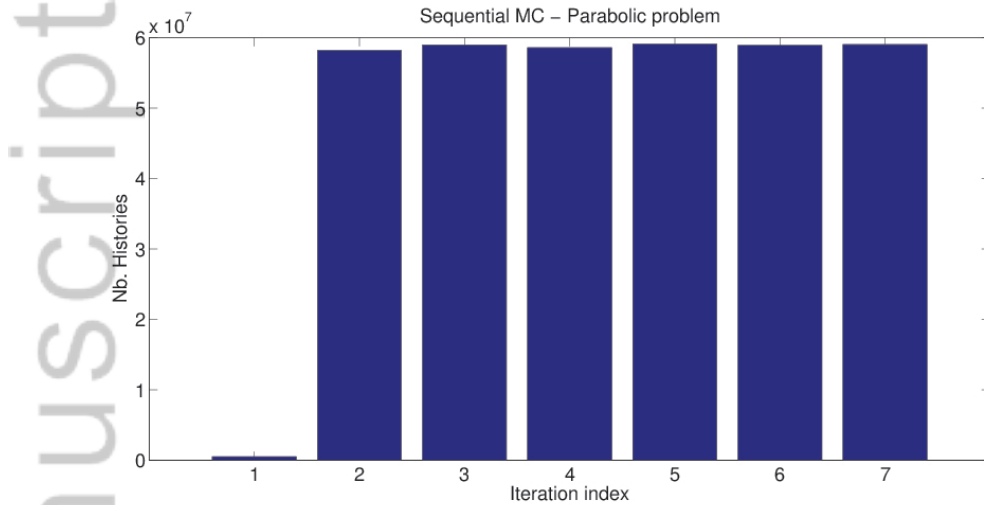


Figure 5. Sequential Monte Carlo - Parabolic problem. Number of random walks employed at each iteration. The number for the first iteration is 505,000.

As before, the threshold for the check on the relative residual is set to  $\varepsilon = 10^{-8}$ , while the threshold for the adaptive selection of the random walks is set to  $\varepsilon_1 = 0.1$ . The results for all three methods are shown in Table XVI. As one can see, both Sequential Monte Carlo and MCSA dramatically reduce the number of iterations with respect to the purely deterministic preconditioned Richardson iteration, with MCSA outperforming Sequential Monte Carlo. Of course each iteration is now more expensive due to the Monte Carlo calculations required at each Richardson iteration, but we stress that Monte Carlo is an embarrassingly parallel method. Monte Carlo calculations are also expected to be more robust in the presence of faults, which is one of the main motivations for the present work.

Finally, Figures 5 and 6 show the number of Monte Carlo histories per iteration for Sequential Monte Carlo and for MCSA, respectively.

## 5. CONCLUSIONS AND FUTURE WORK

In this paper we have reviewed known convergence conditions for Monte Carlo linear solvers and established a few new sufficient conditions. In particular, we have determined classes of matrices for which the method is guaranteed to converge. The main focus has been on the recently proposed MCSA algorithm, which clearly outperforms previous approaches. This method combines a deterministic fixed point iteration (preconditioned Richardson method) with a Monte Carlo acceleration scheme, typically an Adjoint Monte Carlo estimator. Various types of preconditioners have been tested, including diagonal, block diagonal, and sparse approximate inverse preconditioning.

Generally speaking, it is difficult to ensure a priori that the hybrid solver will converge. In particular, convergence of the underlying preconditioned Richardson iteration is necessary, but not sufficient. The application of hybrid solvers to non-diagonally dominant, steady-state problems

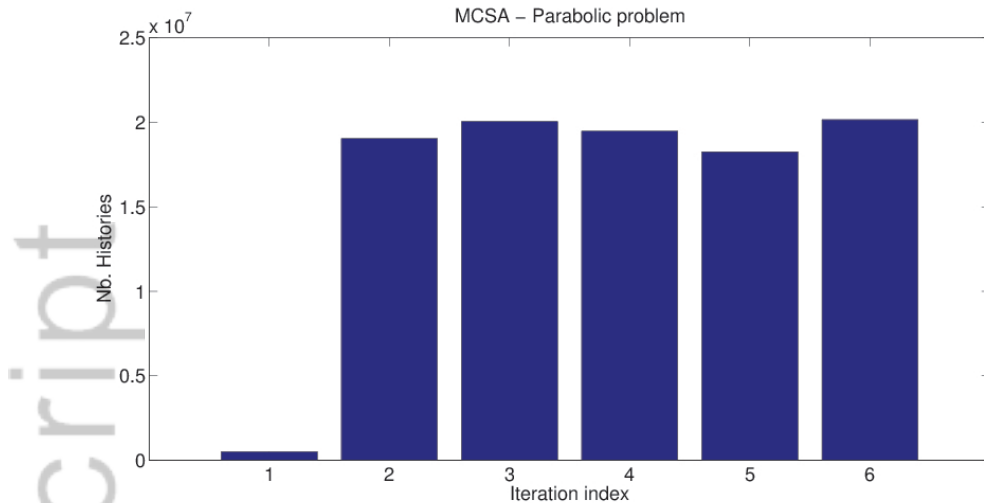


Figure 6. MCSA - Parabolic problem. Number of random walks employed at each iteration. The number for the first iteration is 505,000.

presents a challenge and may require some trial-and-error in the choice of tuning parameters, such as the block size or drop tolerances; convergence can be guaranteed for some standard model problems but in general it is difficult to enforce. This is an inherent limitation of hybrid deterministic-stochastic approaches of the kind considered in this paper. It is of course possible in many cases to obtain convergence by combining the MCSA algorithm with a flexible Krylov subspace method like FGMRES. However, this entails additional costs, decreased parallelism and increased storage requirements, as well as a possible reduction in the resilience of the algorithm due to the need for orthogonalization and the attendant additional communication needed.

On a positive note, numerical experiments show that these methods are quite promising for solving strictly diagonally dominant linear systems arising from time-dependent simulations, such as unsteady diffusion and advection-diffusion type equations. Problems of this type are quite important in practice, as they are often the most time-consuming part of many large-scale CFD and radiation transport simulations. Linear systems with such properties also arise in other application areas, such as network science and data mining.

In this paper we have not attempted to analyze the algorithmic scalability of the hybrid solvers. A difficulty is the fact that these methods contain a number of tuneable parameters, each one of which can have great impact on performance and convergence behavior: the choice of preconditioner, the stopping criteria used for the Richardson iteration, the criteria for the number and length of Monte Carlo histories to be run at each iteration, the particular estimator used, and possibly others. While the cost per iteration is linear in the number  $n$  of unknowns, it is not clear how to predict the rate of convergence of the outer iterations, since it depends strongly on the amount of work done in the Monte Carlo acceleration phase, which is also not known a priori except for some rather conservative upper bounds. Clearly, the scaling behavior of hybrid methods with respect to problem size needs to be further investigated.

Future work should also focus on testing hybrid methods on large parallel architectures and on evaluating their resiliency in the presence of simulated faults. Efforts in this direction are currently under way.

#### ACKNOWLEDGEMENTS

This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up,

irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

We would also like to thank Miroslav Tůma for providing the AINV code used in some of the numerical experiments.

#### REFERENCES

1. E. AGULLO, L. GIRAUD, A. GUERMOUCHE, J. ROMAN, AND M. ZOUNON, *Towards resilient parallel linear Krylov solvers: recover-restart strategies*, Research Report N. 8324, June 2013, Project-Teams HiePACS, INRIA.
2. V. ALEXANDROV, E. ATANASSOV, I. T. DIMOV, S. BRANFORD, A. THANDAVAN, AND C. WEIHRAUCH, *Parallel hybrid Monte Carlo algorithms for matrix computations problems*, Lecture Notes in Computer Science, 3516 (2005), pp. 752–759.
3. O. AXELSSON, *Iterative Solution Methods*, Cambridge University Press, 1994.
4. M. BENZI, *Preconditioning techniques for large linear systems: a survey*, Journal of Computational Physics, 182 (2002), pp. 417–477.
5. M. BENZI, C. D. MEYER, AND M. TUMA, *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM Journal on Scientific Computing, 17 (1996), pp. 1135–1149.
6. M. BENZI AND M. TŮMA, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM Journal on Scientific Computing, 19 (1998), pp. 968–994.
7. A. BERMAN AND R. J. PLEMMONS, *Nonnegative Matrices in the Mathematical Sciences*, Academic Press, New York, 1979.
8. R. COURANT, K. FRIEDRICHS, AND H. LEWY, *Über die partiellen Differenzgleichungen der mathematischen Physik*, Mathematische Annalen, 100 (1928), pp. 32–74.
9. J. H. CURTISS, *Sampling methods applied to differential and difference equations*, Proc. Seminar on Scientific Computation, Nov. 1949, IBM, New York, 1950, pp. 87–109.
10. I. T. DIMOV AND V. N. ALEXANDROV, *A new highly convergent Monte Carlo method for matrix computations*, Mathematics and Computers in Simulation, 47 (1998), pp. 165–181.
11. I. T. DIMOV, V. ALEXANDROV, AND A. KARAIVANOVA, *Parallel resolvent Monte Carlo algorithms for linear algebra problems*, Mathematics and Computers in Simulation, 55 (2001), pp. 25–35.
12. H. C. ELMAN, A. RAMAGE, AND D. J. SILVESTER, *IFISS: A computational laboratory for investigating incompressible flow problems*, SIAM Review, 56 (2014), pp. 261–273.
13. T. M. EVANS, S. W. MOSHER, S. R. SLATTERY, AND S. P. HAMILTON, *A Monte Carlo synthetic-acceleration method for solving the thermal radiation diffusion equation*, Journal of Computational Physics, 258 (2014), pp. 338–358.
14. M. FASI, J. LANGOU, Y. ROBERT, AND B. UÇAR, *A backward/forward recovery approach for the preconditioned conjugate gradient method*, arXiv:1511.04478v1, 13 November 2015. To appear in Journal of Computational Science.
15. D. F. FEINGOLD AND R. S. VARGA, *Block diagonally dominant matrices and generalizations of the Gerschgorin circle theorem*, Pacific Journal of Mathematics, 4 (1962), pp. 1241–1250.
16. G. E. FORSYTHE AND R. A. LEIBLER, *Matrix inversion by a Monte Carlo method*, Math. Tables Other Aids Comput., 6 (1952), pp. 78–81.
17. S. FRIEDLAND AND S. KARLIN, *Some inequalities for the spectral radius of non-negative matrices and applications*, Duke Mathematical Journal, 42 (1975), pp. 459–490.
18. J. H. HALTON, *Sequential Monte Carlo*, Mathematical Proceedings of the Cambridge Philosophical Society, 58 (1962), pp. 57–58.
19. J. H. HALTON, *Sequential Monte Carlo techniques for the solution of linear systems*, Journal of Scientific Computing, 9 (1994), pp. 213–257.
20. J. HAO, M. MASCAGNI, AND Y. LI, *Convergence analysis of Markov chain Monte Carlo linear solvers using Ulam-von Neumann algorithm*, SIAM Journal on Numerical Analysis, 51 (2013), pp. 2107–2122.
21. M. HOEMMEN AND M. HEROUX, *Fault-tolerant iterative methods via selective reliability*, Proceedings of the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC), vol. 3, IEEE Computer Society, 2011.
22. R. A. HORN AND C. R. JOHNSON, *Topics in Matrix Analysis*, Cambridge University Press, 1994.
23. L. LI, *On the iterative criterion for generalized diagonally dominant matrices*, SIAM Journal on Matrix Analysis and Applications, 24 (2002), pp. 17–24.
24. Y. SAAD, *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, Philadelphia, 2003.
25. K. SARGSYAN, F. RIZZI, P. MYCEK, C. SAFTA, K. MORRIS, H. NAJM, O. LE MAITRE, O. KNIO, AND B. DEBUSSCHERE, *Fault resilient domain decomposition preconditioner for PDEs*, SIAM Journal on Scientific Computing, 37 (2015), pp. A2317–A2345.
26. S. R. SLATTERY, *Parallel Monte Carlo Synthetic Acceleration Methods For Discrete Transport Problems*, Ph.D. Thesis, University of Wisconsin-Madison, 2013, <http://search.proquest.com/ebrary/docview/1449206072>.
27. S. R. SLATTERY, T. M. EVANS, AND P. P. H. WILSON, *A spectral analysis of the domain decomposed Monte Carlo method for linear systems*, International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering, 2013.
28. A. SRINIVASAN, *Monte Carlo linear solvers with non-diagonal splitting*, Mathematics and Computer in Simulation, 80 (2010), pp. 1133–1143.
29. M. STOYANOV AND C. WEBSTER, *Numerical analysis of fixed point algorithms in the presence of hardware faults*, SIAM Journal on Scientific Computing, 37 (2015), pp. C532–C553.