

MONTE CARLO METHODS TO SOLVE SPARSE
LINEAR SYSTEMS

Massimiliano Lupo Pasini

Contents

1	Formulation of plain Monte Carlo linear solvers	3
1.1	Forward Method	4
1.1.1	Stopping criteria	5
1.2	Adjoint Method	5
1.2.1	Stopping criteria	6
1.3	A theoretical comparison between the two methods	6
2	Statistical requirements	8
2.1	Forward method - Expected value and variance of the estimator	8
2.2	Adjoint method - Expected value and variance of the estimator	9
2.3	Numerical interpretation of the statistical requirements	9
2.4	Necessary conditions, sufficient conditions and choice of the transition matrix	10
3	Adaptive approaches	12
3.1	An adaptive cut-off for the length of random walks	12
3.1.1	Numerical results	12
3.2	An adaptive selection of the number of histories	15
3.2.1	Relative standard deviation - Forward method	15
3.2.2	Relative standard deviation - Adjoint method	19
3.2.3	An a posteriori variance-based adaptivity	19
3.3	Numerical results	20
3.4	A stationarity check for the a posteriori adaptivity	23
3.5	An a priori adaptive approach on the residual	24
4	Linear solvers with hybrid schemes	28
4.1	Sequential Monte Carlo	28
4.2	Monte Carlo Synthetic Acceleration	28
4.3	Preconditioning techniques	29
4.3.1	Block Preconditioning	29
4.3.2	Approximate Inverse of a Sparse Matrix (AINV)	30
4.3.3	Incomplete LU Factorization Preconditioners (ILU)	31
4.4	Reordering techniques	32
4.5	Numerical results	32
4.5.1	Illustrative examples	32
4.5.2	SP_N equations	32
5	Applicable sets for MC solvers	37
5.1	Strictly diagonally dominant matrices	37
5.1.1	Numerical results	38
5.2	Generalized diagonally dominant matrices (GDDM)	39
5.3	"Block-diagonally dominant"-like matrices	41
5.3.1	Forward method	41
5.3.2	Adjoint method	42
	References	42

Introduction

For a long period the pursuit of a reliable solver for linear and nonlinear algebraic systems has been one of the most important issues in Numerical Analysis. Above all for mathematical problems derived from practical issues by modeling. The mathematical properties of the model (e.g. likely sparsity pattern of the matrix) have urged scientists to look for efficient and pattern-driven solvers, capable to cope with curse of dimensionality and high scalability requests.

As scalability computing move towards exascale facilities (which stands for machines computing up to 10^{18} flops), new numerical schemes suitable for this kind of devices are strongly demanded. In fact the purpose is to combine high fidelity of the results with the possibility to exploit the hardware enrichment that computer industry is going to provide us with.

Recently new algorithms that combine statistical and numerical devices have been developed. Even though *numerical efficiency* has been sacrificed sometimes, the advantages in terms of *robustness* of these methods make it worth carrying on studies in this direction.

This report is written in order to show some results concerning the resolution of linear systems via a Monte Carlo approach. The work presented here is based on what developed so far on this topic by some of the authors who in literature gave a contribution (see [Hal62], [Hal94], [DA98], [DAK01], [AAD⁺05], [ESW13] and [EMSH14]).

In Chapters 1 the main issue is a comparison between two different approaches, both aiming at solving an algebraic problem stochastically. These two approaches, one transversal to the other, are named *Forward* and *Adjoint Method*. The basic idea is to generate random walks with random variables associated with them. The role of the random variables is to reconstruct the values attained by the solution vector to the linear problem we are interested in.

In Chapter 2 theoretical constraints are presented. These must be respected for convergence issues, referring to results published in [HMY13]. Sufficient conditions, necessary conditions and equivalence conditions for convergence are stated.

Chapter 3 is dedicated to issues associated with implementation and computation. In particular an adaptive approach for the cut-off of the random walks is presented, based on what is described in [Sla13]. Afterwards an adaptive selection of histories to employ is introduced. Basic numerical tests are employed for illustrative purposes.

Chapter 4 is focused on numerical schemes elaborated through the years in order to speed-up the convergence of plain Monte Carlo linear solvers. Specifically formulations of *Sequential Monte Carlo* and *Monte Carlo Synthetic Acceleration* are treated.

Chapter 5 is targeted to combine all the topics covered in the previous chapters to come up with an overview about the state-of-the-art plight.

Chapter 1

Formulation of plain Monte Carlo linear solvers

As already said, the goal of this family of methods is to use random walks for estimating the solution vector of a linear system by interpreting it as the expected value of a random variable (the estimator). The initial probability and the transition matrix are built using the components of the matrix of the system to be solved.

As underlined in [DA98], Monte Carlo methods may be divided into two classes: *direct methods* ([Hal62], [Hal94], [DA98], [DAK01]) and *iterative* ones ([ESW13] and [EMSH14]). The first are characterized by a merely stochastic scheme, therefore the provided error with respect to the exact solution is made of just a stochastic component. The second ones, instead, combine numerical and statistical schemes and they generate two types of error: one is *stochastic*, the other one is *systematic*. It does not mean that it will be always simple to recognize them separately, but it is important to be aware of this intrinsic structure in order to target what is the part of the scheme that requires a finer refinement (e.g. increase of numerical iterations rather than random walks).

We start our treatise by introducing the direct methods.

Assume to start from a linear system such as

$$A\mathbf{x} = \mathbf{b}, \quad (1.1)$$

where $A \in \mathbb{R}^{n \times n}$, $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$.

By applying a left preconditioning, the system (1.1) gets the following shape:

$$P^{-1}A\mathbf{x} = P^{-1}\mathbf{b}, \quad (1.2)$$

1.2 can be reinterpreted as a fixed point scheme

$$\mathbf{x} = H\mathbf{x} + \mathbf{f}. \quad (1.3)$$

where $H = I - P^{-1}A$ and $\mathbf{f} = P^{-1}\mathbf{b}$.

Assuming that the spectral radius $\rho(H) < 1$, the solution to (1.3) can be written in terms of a power series of H (Neumann-Ulam series)

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

Therefore the fixed point scheme generates a sequence of approximate solutions $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ which 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 have

$$x_i = \sum_{\ell=0}^{\infty} \sum_{k_1=1}^n \sum_{k_2=1}^n \cdots \sum_{k_{\ell}=1}^n H_{k_0, k_1} H_{k_1, k_2} \cdots H_{k_{\ell-1}, k_{\ell}} f_{k_{\ell}}. \quad (1.4)$$

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 characterized by the set of indexes of the forcing term \mathbf{f} .

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

At each i -th step of the random walk it is associated a random variable k_i . Its realization represents the index of the component of \mathbf{f} which is visited in the current step of the random walk.

The way the transition probability is built and the way the initial state of the random walk is chosen give birth to the two different approaches we are going to use.

1.1 Forward Method

The goal is to evaluate a functional such as

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

where $\mathbf{h} \in \mathbb{R}^n$ is the Riesz representative in \mathbb{R}^n of the functional J . We can use it to build the initial probability $\tilde{p} : S \rightarrow [0, 1]$ of the random walk such that

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

It is important to stress out that the role of vector \mathbf{h} is restricted to the building of the initial probability. Once that is done, it is not used anymore in the definition of the stochastic process. A possible choice for the transition probability P instead can be

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

where $pr(\cdot, i) : S \rightarrow [0, 1] \forall i \in S$. A related sequence of random variables $w_{i,j}$ can be defined such that

$$w_{i,j} = \frac{H_{i,j}}{P_{i,j}}.$$

The probability distribution of the random variables $w_{i,j}$ is represented by the transition matrix that rules the stochastic process. The sequence of random variables just introduced can be used to build one more sequence of random variables, whose expected value will assume the role of estimator. At first we introduce quantities W_j such that

$$W_0 = \frac{h_{k_0}}{\tilde{p}_{k_0}}, \quad W_j = W_{j-1} w_{i,j}, \quad j = 1, \dots, i.$$

By defining

$$X_i(\nu) = \sum_{m=0}^k W_m f_{i_m}$$

as the random variable associated with a specific permutation ν of the random walk indexed by i , we can introduce the estimator θ_i such as

$$\theta_i = E[X_i] = \sum_{\nu} P_{\nu} X_i(\nu), \quad i = 1, \dots, n.$$

Integer n represents the size of the solution vector to 1.1 and index i is referred to the component of the solution vector we want to compute. P_{ν} is the probability associated with a specific permutation of the random walk. It can be proved that

$$E[W_i f_{k_i}] = (\mathbf{h}, H^i \mathbf{f}), \quad i = 0, 1, 2, \dots$$

and

$$\theta_i = E \left[\sum_{i=0}^{\infty} W_i f_{k_i} \right] = (\mathbf{h}, \mathbf{x}).$$

A possible choice for \mathbf{h} is a vector of the canonical basis. This would correspond in setting manually the initial state of the random walk because the related initial probability is a Dirac delta

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

By doing so, we have

$$\theta_i = E \left[\sum_{\ell=0}^{\infty} W_{\ell} f_{k_{\ell}} \right] = x_i = \sum_{\ell=0}^{\infty} \sum_{k_1=1}^n \sum_{k_2=1}^n \cdots \sum_{k_{\ell}=1}^n P_{k_0,k_1} P_{k_1,k_2} \cdots P_{k_{\ell-1},k_{\ell}} w_{k_0,k_1} w_{k_1,k_2} \cdots w_{k_{\ell-1},k_{\ell}} f_{k_{\ell}}. \quad (1.5)$$

The idea sitting behind is to fix a priori a certain number of random walks, which will be sampled for each component of the vector \mathbf{x} . This is strictly correlated to the fact that we can decide what is the functional of the solution we are interested in computing. Thus we can take the Dirac delta as a functional and it implies to select the initial state for all the random walks.

1.1.1 Stopping criteria

In equation (1.5) we have an infinite sum, which obviously needs to be stopped at a finite index in the computational phase. Here it is chosen to fix a priori the maximal number of steps characterizing each random walk. Therefore we will consider

$$\tilde{\theta}_i = E \left[\sum_{\ell=0}^m W_{\ell} b_{k_{\ell}} \right] = x_i = \sum_{\ell=0}^m \sum_{k_1=1}^n \sum_{k_2=1}^n \cdots \sum_{k_{\ell}=1}^n P_{k_0,k_1} P_{k_1,k_2} \cdots P_{k_{\ell-1},k_{\ell}} w_{k_0,k_1} w_{k_1,k_2} \cdots w_{k_{\ell-1},k_{\ell}} f_{k_{\ell}}.$$

A different criterion that is more reasonable to use is to fix a cut-off threshold ε on the estimator W_i , so that the random walk is stopped at the m -th step if $W_m < \varepsilon$. Other possibilities for the cut-off are discussed in [Hal94].

1.2 Adjoint Method

A second Monte Carlo method can be derived by considering the linear system adjoint to (1.2)

$$(P^{-1}A)^T \mathbf{y} = \mathbf{d}, \quad (1.6)$$

where \mathbf{y} and \mathbf{d} are the adjoint solution and source term. Instead $(P^{-1}A)^T$ is the transpose matrix (previously preconditioned). By rewriting (1.6) in a fixed point approach, we have

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

The Neumann-Ulam series now reads

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

By expanding the summation for each component of the solution vector \mathbf{y} , as we did for the Forward Method, we have

$$y_i = \sum_{\ell=0}^{\infty} \sum_{k_1=1}^n \sum_{k_2=1}^n \cdots \sum_{k_{\ell}=1}^n H_{k_{\ell},k_{\ell-1}} \cdots H_{k_2,k_1} H_{k_1,k_0} d_{k_{\ell}}. \quad (1.7)$$

The formula to compute the solution to the adjoint problem can be exploited in order to compute the solution to the primal one. This is possible by properly fixing the additional degrees of freedom that characterizes the adjoint problem, which is the arbitrary forcing term. Indeed by setting

$$\mathbf{d} = \delta_j$$

we have

$$(\mathbf{y}, \mathbf{f}) = (\mathbf{x}, \delta_j) = x_j.$$

The purpose we want to achieve, in this case, is the computation of the functional given by the scalar product

$$J^*(\mathbf{y}) = (\mathbf{b}, \mathbf{y}).$$

In this case the Riesz representative is the source term \mathbf{f} of the primal problem 1.2. Differently from what done in the Forward Method, here the initial step of the random walk cannot be chosen arbitrarily. In fact the Riesz representative now is something fixed a priori. Therefore we must define the initial probability as

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

The transition matrix is defined as

$$P_{i,j} = \frac{|H_{i,j}^T|}{\sum_{k=1}^n |H_{i,k}^T|} = \frac{|H_{j,i}|}{\sum_{k=1}^n |H_{k,i}|}$$

Analogously for the random variables

$$w_{i,j} = \frac{H_{j,i}}{P_{i,j}}.$$

Variables W_i are determined recursively as before. By computing the expected value for the estimator in this case we have

$$\theta_j = E \left[\sum_{\ell=0}^{\infty} W_{\ell} \delta_{k_{\ell}, j} \right] = \sum_{\ell=0}^{\infty} \sum_{k_1}^n \sum_{k_2}^n \cdots \sum_{k_{\ell}}^n b_{k_0} P_{k_0, k_1} P_{k_1, k_2} \cdots P_{k_{\ell-1}, k_{\ell}} w_{k_0, k_1} \cdots w_{k_{\ell-1}, k_{\ell}} \delta_{k_{\ell}, j}. \quad (1.8)$$

We can notice now that the component of the source vector, in each permutation of the random walk, is associated with the initial step. The delta at the end of the series stands for a filter. It means that if I want to estimate the j -th component of the solution vector, just a subset of all the random walks gives a contribution to this. In particular, just those random walks that currently reside on the index I want to estimate are considered. In other words, we can say that all the random walks give contribution to determine the entire solution vector, since at each step of the random walk the estimator is used to refine the estimation of the component associated with the current index. In the following step the state visited by the random walk will change, thus the new value of the estimator will contribute to refine the estimation of another component of the solution vector.

1.2.1 Stopping criteria

Also for the adjoint method we have implemented the stopping criterion described as follows

$$\tilde{\theta}_j = E \left[\sum_{\ell=0}^m W_{\ell} \delta_{k_{\ell}, j} \right] = \sum_{\ell=0}^m \sum_{k_1}^n \sum_{k_2}^n \cdots \sum_{k_{\ell}}^n b_{k_0} P_{k_0, k_1} P_{k_1, k_2} \cdots P_{k_{\ell-1}, k_{\ell}} w_{k_0, k_1} \cdots w_{k_{\ell-1}, k_{\ell}} \delta_{k_{\ell}, j}.$$

However it may be more reasonable to use the weight cut-off we have already discussed for the Forward method. For sure we will use this in the future.

1.3 A theoretical comparison between the two methods

Both the forward and the adjoint methods are characterized by a low convergence rate, due to the Central Limit Theorem (CLT). If we refer to N as the total number of random walks used to approximate a component of \mathbf{x} for the forward method, we know that

$$x_i - E \left[\sum_{\ell=0}^{\infty} W_{\ell} f_{k_{\ell}} \right] = \mathcal{O} \left(\frac{1}{\sqrt{N}} \right).$$

In the case of the adjoint method instead, N is conceived as the total number of random walks used to evaluate the entire solution vector. In this case we have

$$\mathbf{x} - E \left[\sum_{l=0}^{\infty} W_l \delta_{k_l, j} \right] = \mathcal{O} \left(\frac{1}{\sqrt{N}} \right).$$

The adjoint method is kind of a cheaper algorithm, because it exploits each random walk in order to maximize the amount of information we can extract from it. This implies that, fixed N , the solution provided by the forward method is more accurate, since with the forward method N random walks are used for each component.

The advantages implied by using these stochastic approaches concern the opportunity to parallelize the code. In fact the schemes introduced above are highly scalable. Since each random walk is stochastically independent from all the others, each one of them can be attributed to a specific processor. All the nodes can work independently and results coming from all the random walks can be collected just when all the nodes are done, in order to compute the mean value of the estimator.

As it can be notice from (1.5) and (1.8), these formulas are the expansion of the Neumann series. Therefore Monte Carlo Forward Method and Monte Carlo Adjoint method are expected to decrease the time required for convergence, as the spectral radius of their iteration matrix decreases. Of course this observation assumes that the weight cut-off is used as a stopping criterion.

Chapter 2

Statistical requirements

For the sake of completeness and intelligibility of what will be introduced later on, we report here the statement of the Central Limit Theorem.

Theorem 1 Central Limit Theorem (CLT) for independent sequences

Suppose X_1, X_2, \dots is a sequence of independent identically distributed (i.i.d.) random variables with $E[X_i] = \mu$ and $\text{Var}[X_i] = \sigma^2 < \infty$. Then as $n \rightarrow \infty$, the random variables $\sqrt{n}(S_n - \mu)$ converge in distribution to a normal $\mathcal{N}(0, \sigma^2)$:

$$\sqrt{n} \left(\left(\frac{1}{n} \sum_{i=1}^n X_i \right) - \mu \right) \xrightarrow{d} \mathcal{N}(0, \sigma^2).$$

This is the reference point for what is introduced from now on. In particular the goal is to define a proper transition probability, such that the hypotheses of the theorem hold.

2.1 Forward method - Expected value and variance of the estimator

As already described in Section 1.1 of Report 1, the Forward method is characterized by the introduction of proper weight W_i defined recursively in terms of the steps of the random walk. If these weights get too small, there is no point in keeping the random walk moving, so it is possible to truncate it without significant loss of accuracy.

The formula for the estimator of the Forward Monte Carlo for a generic entry of the solution vector has already been defined in (1.5).

As regards the variance, we know that the following relation holds:

$$\text{Var} \left[\sum_{\ell=0}^{\infty} W_{\ell} f_{k_{\ell}} \right] = E \left[\sum_{\ell=0}^{\infty} W_{\ell}^2 f_{k_{\ell}}^2 \right] - \left(E \left[\sum_{\ell=0}^{\infty} W_{\ell} f_{k_{\ell}} \right] \right)^2. \quad (2.1)$$

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

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

$$E \left[\sum_{\ell=0}^{\infty} W_{\ell} f_{k_{\ell}} \right] < \infty \quad (2.2)$$

$$E \left[\sum_{\ell=0}^{\infty} W_{\ell}^2 f_{k_{\ell}}^2 \right] < \infty \quad (2.3)$$

2.2 Adjoint method - Expected value and variance of the estimator

As concerns the Adjoint method the formula for the expected value of the estimator is represented in (1.8).

The variance is

$$\text{Var} \left[\sum_{\ell=0}^{\infty} W_{\ell} f_{k_0} \delta_{k_{\ell},j} \right] = E \left[\sum_{\ell=0}^{\infty} W_{\ell}^2 f_{k_0}^2 \delta_{k_{\ell},j} \right] - \left(E \left[\sum_{\ell=0}^{\infty} W_{\ell} f_{k_0} \delta_{k_{\ell},j} \right] \right)^2 \quad j = 1, \dots, n. \quad (2.4)$$

On the same lead of what we did for the Forward method, we have to impose finiteness of the expected value and second moment as well. Therefore we require the following

$$E \left[\sum_{\ell=0}^{\infty} W_{\ell} \delta_{k_{\ell},j} \right] < \infty \quad j = 1, \dots, n \quad (2.5)$$

and

$$E \left[\sum_{\ell=0}^{\infty} W_{\ell}^2 f_{k_0}^2 \delta_{k_{\ell},j} \right] < \infty \quad j = 1, \dots, n. \quad (2.6)$$

2.3 Numerical interpretation of the statistical requirements

The requests imposed on the expected value and second moment of the estimators can be reformulated in a purely deterministic setting by looking at the spectral radius of some properly defined matrices.

For instance the condition of finiteness of the expected value can be reformulated by requiring

$$\rho(H) < 1, \quad (2.7)$$

where H is the iteration matrix of the fixed point scheme. In fact both equations 1.5 and 1.8 can be considered as power series in terms of H . Moreover having a spectral radius of H smaller than 1 is a necessary and sufficient condition for the Neumann series to converge.

By analogy we aim at reproducing a similar reasoning for the second moment of the estimator, since we want it to be finite as well. By looking at the equations of the variance 2.1 and 2.4 for the Forward and the Adjoint method respectively, we notice that the second moment can be reinterpreted as a power series with respect to the matrices defined as follows:

$$H_{i,j}^* = \frac{H_{i,j}^2}{P_{i,j}} \quad \text{- Forward Method.}$$

or

$$H_{i,j}^* = \frac{H_{j,i}^2}{P_{i,j}} \quad \text{- Adjoint Method.}$$

Accordingly to the necessary and sufficient condition for the power series to converge, we require

$$\rho(H^*) < 1. \quad (2.8)$$

Therefore condition 2.7 and 2.8 are the numerical reformulation of the finiteness of the expected value and second moment of the random variables considered. These results can be found in [HMY13].

In the following chapters we will show some examples where the conditions for convergence are not expected, with associated plots to illustrate the stagnation of the error.

2.4 Necessary conditions, sufficient conditions and choice of the transition matrix

In this chapter we get down to enunciate some results presented in [HMY13] about necessary conditions and sufficient conditions in order to reach convergence with a Monte Carlo linear solver. In particular we discuss what is a suitable choice for constructing a transition probability that makes the MC solver converge.

We remember that the construction of the transition probability must satisfy some constraints such as

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

plus one more restriction which varies depending on the particular Monte Carlo approach chosen

Forward Method: $H_{i,j} \neq 0 \Rightarrow P_{i,j} \neq 0$

Adjoint Method: $H_{j,i} \neq 0 \Rightarrow P_{i,j} \neq 0$

The previous restrictions on the attainable values for the entries of the transition probability are called *transition conditions*.

We start by presenting

Theorem 2 Consider a vector $a = (a_1, a_2, \dots, a_N)^T$ where at least one element is nonzero, $a_k \neq 0$ for some $k \in \{1, \dots, N\}$.

- For a probability vector $p = (p_1, p_2, \dots, p_N)^T$ satisfying the transition conditions, the lower bound of $\sum_{k=0}^N \frac{a_k^2}{p_k}$ is $\left(\sum_{k=1}^N |a_k| \right)^2$
- There always exists a probability vector such that $\sum_{k=0}^N \frac{a_k^2}{p_k} \geq c \geq 1$ for all $c > 1$.

It might be observed that the minimum of the quantity $\sum_{k=0}^N \frac{a_k^2}{p_k}$ is attained when the probability vector is defined such as $p_k = \frac{|a_k|}{\sum_{k=1}^N |a_k|}$. It implies that the MAO (Monte Carlo Almost Optimal) probability is the one that reduces as much as possible the ∞ -norm of H^* .

For the sake of simplicity let us introduce now a generic random walk truncated at a certain k -th step

$$\gamma_k : r_0 \rightarrow r_1 \rightarrow r_2 \rightarrow \dots \rightarrow r_k$$

and the statistical estimator

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

which is associated with the Forward Method.

Then the following theorem holds

Theorem 3 (Suited to the Forward Method)

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

If we introduce the estimator

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

which is associated with the Adjoint Method, then we can state a theorem specular to 3:

Theorem 4 (*Suited to the Adjoint Method*)

Let $H \in \mathbb{R}^{n \times n}$ with $\|H\|_1 < 1$. Consider ν_k as the realization of a random walk γ 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 $k \rightarrow \infty$.

In particular, relying on what guaranteed by Theorem 2, in both Theorems 3 and 4 the condition holds for sure with the MAO transition matrix.

These theoretical results represent sufficient condition for the Forward and Adjoint Monte Carlo to converge and can be easily checked. However requiring $\|H\|_{\infty} < 1$ rather than $\|H\|_1 < 1$ may be too demanding. There exist some preconditioners that guarantee that this condition be respected for sure (e.g. approximate inverse preconditioners with a proper value of dropping tolerance). Nevertheless it may require the loss of sparsity of the iteration matrix and of course this is something we cannot accept in our numerical setting.

Therefore, in situation where 3 and 4 are too strong conditions to be verified, the necessary and sufficient condition on the spectral radius of H^* is the only criterion to look at. The drawback of the necessary and sufficient condition, however, consists in computing the spectral radius of a matrix. Indeed the computational cost equals the one of a deterministic solver for the resolution of the original linear system we are interested in.

We now provide a condition that, if verified, entails the impossibility of reaching convergence with any Monte Carlo linear solver.

Theorem 5 Let H be an $N \times N$ matrix with spectral radius $\rho(H) < 1$. Let H^+ be the $N \times N$ matrix where $H_{i,j}^+ = |H_{i,j}|$. If $\rho(H^+) > 1$, there does not exist a transition matrix P satisfying the transition conditions such that the variance $\text{Var}(X(\gamma_k))$ converges to zero as $k \rightarrow \infty$.

Theorem 5 hold for both Forward and Adjoint method, since H^+ and $(H^+)^T$ have the same spectral radius.

Chapter 3

Adaptive approaches

3.1 An adaptive cut-off for the length of random walks

At first we focus on the definition of a weight cut-off in order to decide where to terminate the histories. It implies that we are looking for a quantity that tells us how many terms of the Neumann series need to be considered.

The goal is to find an automatic way for the cutoff of the histories. Therefore we are looking for an m such that

$$\tilde{\theta}_i = E \left[\sum_{\ell=0}^m W_{\ell} f_{k_{\ell}} \right] = x_i = \sum_{\ell=0}^m \sum_{k_1=1}^n \sum_{k_2=1}^n \cdots \sum_{k_{\ell}=1}^n P_{k_0, k_1} P_{k_1, k_2} \cdots P_{k_{\ell-1}, k_{\ell}} w_{k_0, k_1} w_{k_1, k_2} \cdots w_{k_{\ell-1}, k_{\ell}} f_{k_{\ell}}.$$

and

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

are good approximations of 1.5 and 1.8 respectively.

In [Sla13] there is a criterion for the cutoff of the random walk that works either for the Forward or for the Adjoint method. It consists in defining a cutoff relative threshold W_c and looking for a step such that

$$W_m \leq W_c W_0. \quad (3.1)$$

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.

When the inequality 3.1 holds, then it means that all the steps of the random walk that come after the m -th one are negligible, since the tally contributions become increasingly small. It is equivalent to say that all the powers of the iteration matrix that are bigger than m can be ignored in the Neumann series.

Of course the point where the random walks are cut does not have to be selected too early in the sequence of steps, otherwise it will compromise the accuracy of the computed solution.

In order to understand the efficiency of this criterion and what might be a reasonable choice for W_c , we tuned the weight cutoff to compute the solution to a set of problems.

3.1.1 Numerical results

In this section we consider the Jacobi preconditioning in order to rewrite the linear system $A\mathbf{x} = \mathbf{b}$ into a fixed point scheme.

The resulting iteration matrix H is such that

$$H = I - D^{-1}A, \quad D = \text{diag}(A).$$

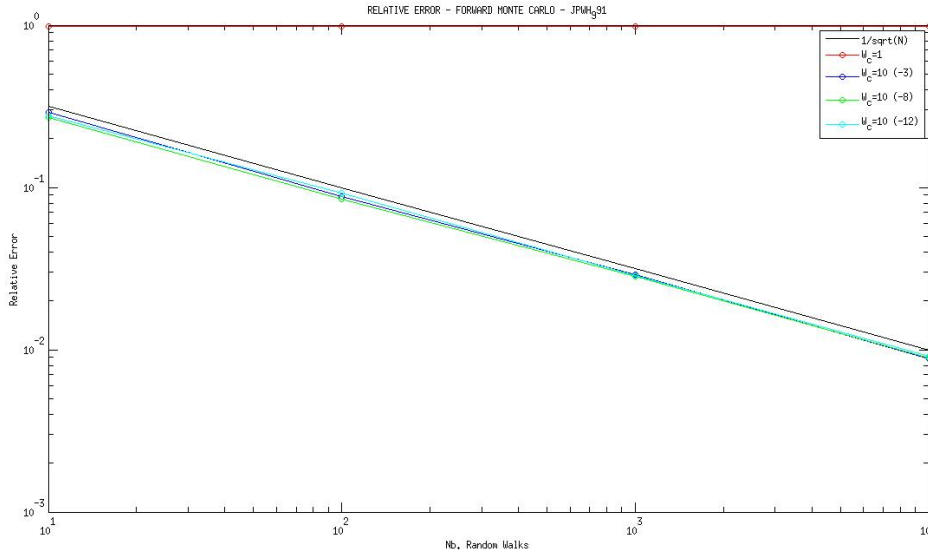


Figure 3.1: Forward MC: JPWH_991 - Relative error with a weight cutoff $W_c = 1$ (red), $W_c = 10^{-3}$ (green), $W_c = 10^{-8}$ (blue) and $W_c = 10^{-12}$ (cyan).

Forward Monte Carlo

For now we focus on the Forward Method with an almost optimal probability to compute the solution to $A\mathbf{x} = \mathbf{b}$ with the standard Monte Carlo algorithm. Different values for the weight cutoff have been used.

In Figure 3.2 we have results for the the Marshak Wave problem (also called as Thermal Equation in this report). The matrix associated with this problem is such that $A \in \mathbb{R}^{1600 \times 1600}$. The spectral radius of H is $\rho(H) = 0.6$, while the spectral radius of H^* is $\rho(H^*) = 0.3758$. Therefore we expect to need a few terms of the Neumann series in order to compute an accurate solution. By setting the cutoff threshold to 1 we have a departure of the relative error from the Central Limit Theorem behavior. This happens because the threshold requires the random walks to be cut too early. Therefore the approximated Neumann series neglects powers of the iteration matrix that should be taken into account. It can be noticed that already for $W_c = 10^{-4}$ we succeed in reproducing the CLT behavior pretty well. However, as the number of employed random walks increases, the descent gets lower. This is due to the fact that, in order to get a more precise solution, we cannot just increase the number of random walks. We have to increase the number of steps per history as well. This is a phenomenon that will be stressed out also by the error behavior associated with other test cases.

Let us consider now the "JPWH_991" matrix. It sits in $\mathbb{R}^{991 \times 991}$ and the associated iteration matrix has a spectral radius $\rho(H) = 0.979$. As concerns the H^* matrix the spectral radius is $\rho(H^*) = 0.9797$. Therefore we expect to need a massive number of steps in order to reproduce the theoretical trend. It turns into the necessity to set W_c to smaller values than the ones used for the Marshak equation. By doing so, the behavior of this matrix is coherent with the CLT, as it can be observed for $W_c = 10^{-3}$. (Figure 3.1).

More illustrative in terms of the usefulness of the tuning of W_c is the 1d shifted Laplacian. This is a 50 by 50 matrix and the spectral radius of the iteration matrix is $\rho(H) = 0.499$. For different values of the threshold, a progressive improvement in terms of error descent is noticed (see Figure 3.3).

In conclusion we present a very difficult test case, the "FS_680_1". The iteration matrix related to it has a spectral radius equal to 0.97. As it will be stressed out also later on in the Report, however, the iteration matrix has a particular sparsity pattern (see Figure 3.7). A lot of rows are full of zeros, causing the sudden death of many random walks in the Forward method. Moreover the spectral radius of H^* is $\rho(H^*) = 1.2554$. Therefore the Monte Carlo estimator has infinite variance. This implies that the linear solver does not converge to provide an accurate estimation of the solution to 1.1. It is actually confirmed by the stagnation of the error in Figure 3.4.

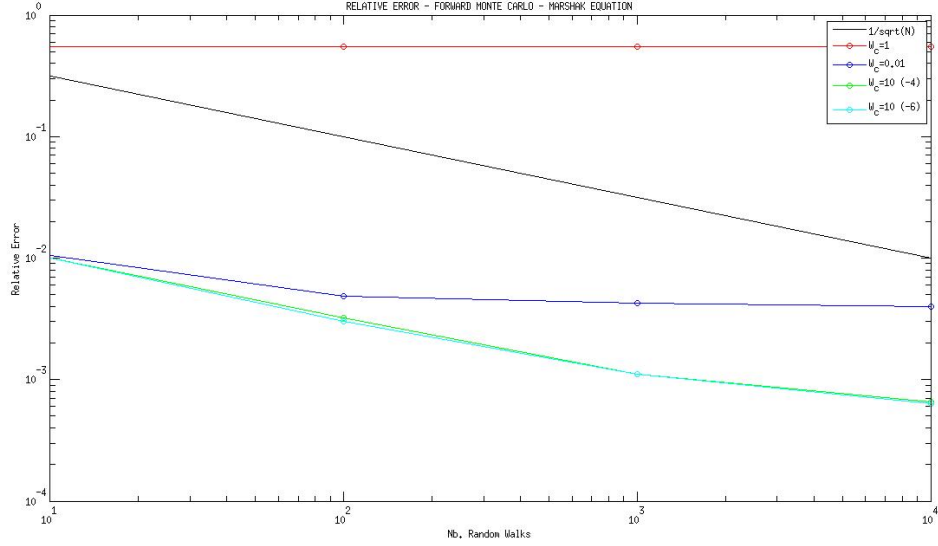


Figure 3.2: Forward MC: Thermal Equation - Relative error with a weight cutoff $W_c = 1$ (red), $W_c = 0.01$ (green), $W_c = 10^{-4}$ (blue) and $W_c = 10^{-6}$ (cyan).

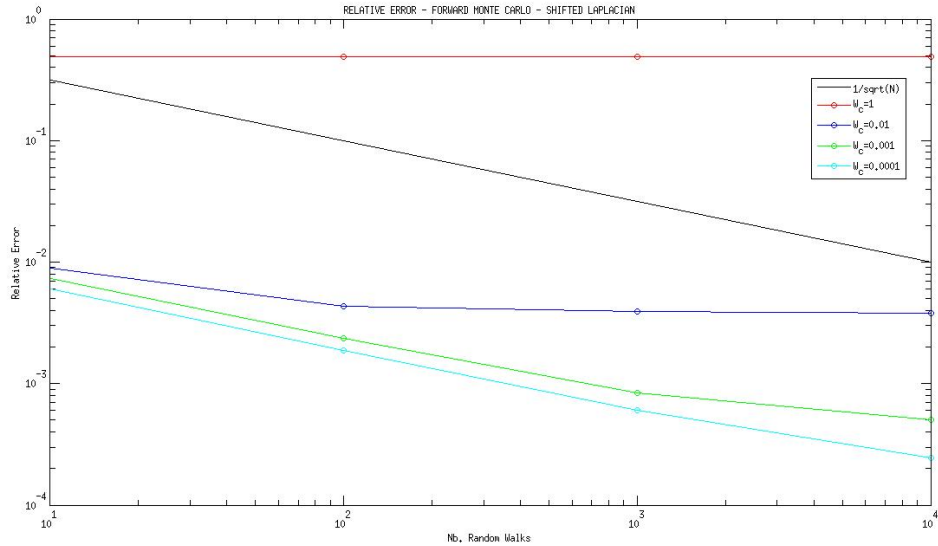


Figure 3.3: Forward MC: Shifted 1D Laplacian - Relative error with a weight cutoff $W_c = 1$ (red), $W_c = 0.01$ (green), $W_c = 0.001$ (blue) and $W_c = 0.0001$ (cyan).

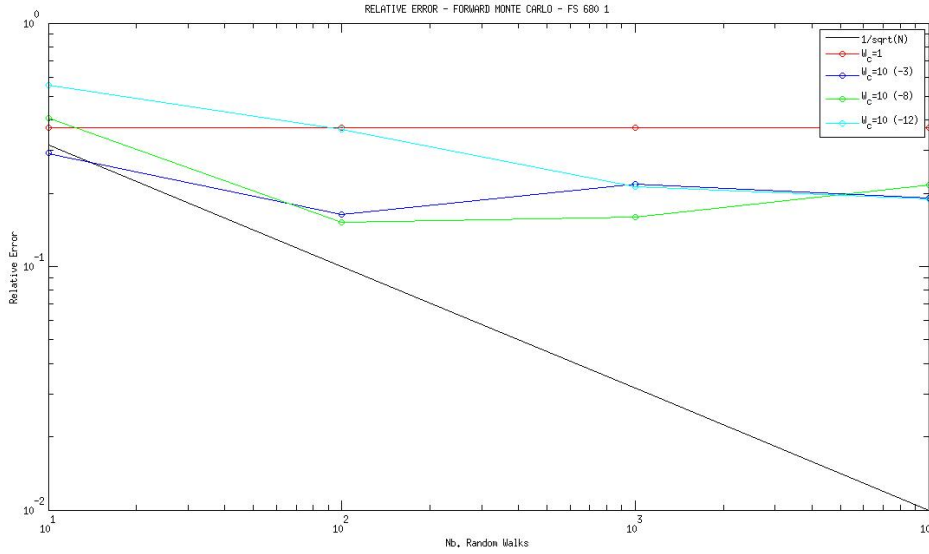


Figure 3.4: Forward MC: FS_680_1 - Relative error with a weight cutoff $W_c = 1$ (red), $W_c = 10^{-3}$ (green), $W_c = 10^{-8}$ (blue) and $W_c = 10^{-12}$ (cyan).

Adjoint Monte Carlo

Now we focus on the Adjoint Method. As concerns the thermal equation, the spectral radius of H^* is $\rho(H^*) = 0.3815$. The relative error with respect to the varying number of employed random walks is shown in Figure 3.5. For $W_c = 1$ we still have a departure from the theoretical trend, which is restored for $W_c = 0.1$ instead restores the CLT behavior, since a larger number of steps for each random walk is preserved by the cutoff.

The "ifiss_advection_diffusion" matrix has a spectral radius for H^* equal to $\rho(H^*) = 0.9827$. By looking at the trend of the error for different cut-off weights in Figure 3.9, we see that $W_c = 0.001$ is already enough to reproduce the behavior of the CLT curve. Similar reasoning hold for the two-dimensional Laplacian.

As concerns "FS_680_1", the convergence of the Monte Carl linear solver is not achieved even by the Adjoint method. In fact $\rho(H^*) = 3.598$. Figure 3.6 shows that the change of threshold for the cutoff does not change the slope of the relative error curve.

A very interesting result is provided by the shifted 1D Laplacian, in terms of cutoff effectiveness. For $W_c = 1$ it is detected a departure of the error from the slope expected by the CLT. However even just for $W_c = 0.1$ the situation gets better and the $\frac{1}{\sqrt{N}}$ trend for the error is restored.

3.2 An adaptive selection of the number of histories

In the previous chapter we have focused on the possibility to cut in an accurate way the random walks. This basing on a threshold that considers the magnitude of the weight at each step.

However the length of a truncated random walk is just one of the two degrees of freedom at hand. Indeed we can also pursue an adaptive way to select a reasonable number of histories to be taken. This is possible for controlling the uncertainty associated with the solution.

In order to do this, we can start from taking into account a statistical quantity that intrinsically quantify the uncertainty related to a random variable: the variance. Therefore the attempt in this case is to control the variance associated with the estimated solution. The goal is to reach a condition where this entails taking over the accuracy of the solution as well. In fact there is a strong correlation between uncertainty and accuracy of an estimation.

3.2.1 Relative standard deviation - Forward method

As regards the Forward method, we know that the expression for the variance is defined in Formula (2.1).

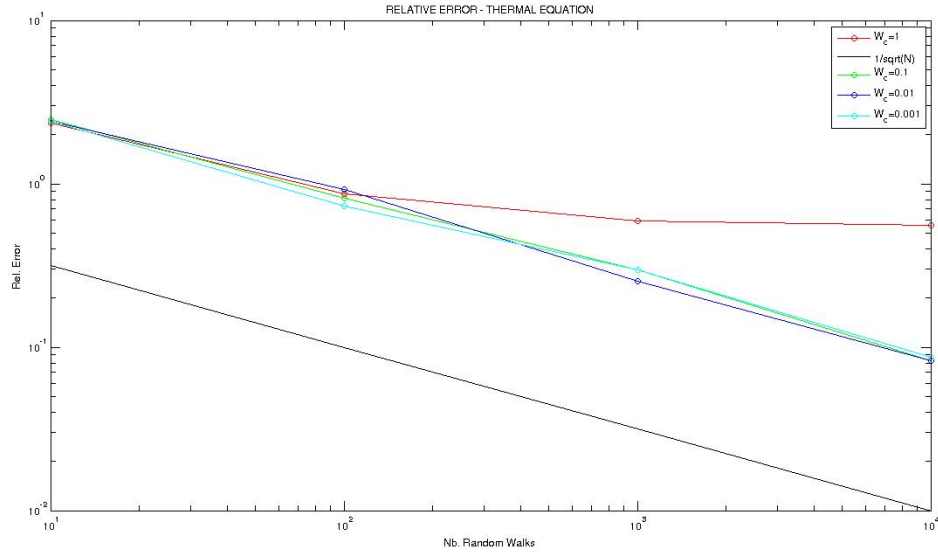


Figure 3.5: Adjoint MC: Thermal Equation - Relative error with a weight cutoff $W_c = 1$ (red), $W_c = 0.1$ (green), $W_c = 0.01$ (blue) and $W_c = 0.001$ (cyan).

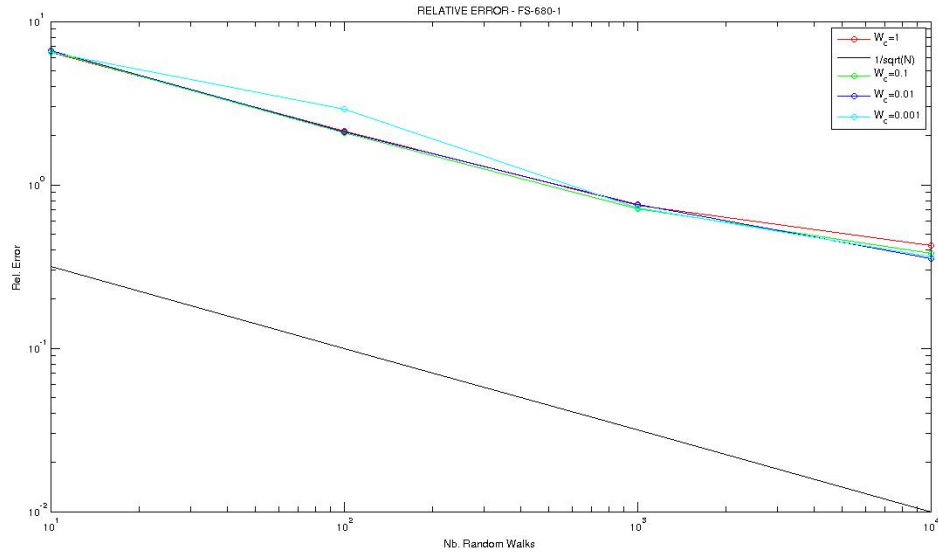


Figure 3.6: Adjoint MC: FS_680_1 - Relative error with a weight cutoff $W_c = 1$ (red), $W_c = 0.1$ (green), $W_c = 0.01$ (blue) and $W_c = 0.001$ (cyan).

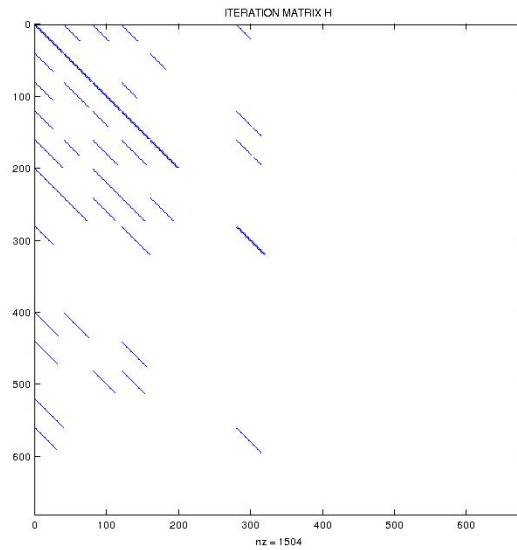


Figure 3.7: FS_680_1 - Sparsity pattern for iteration matrix H .

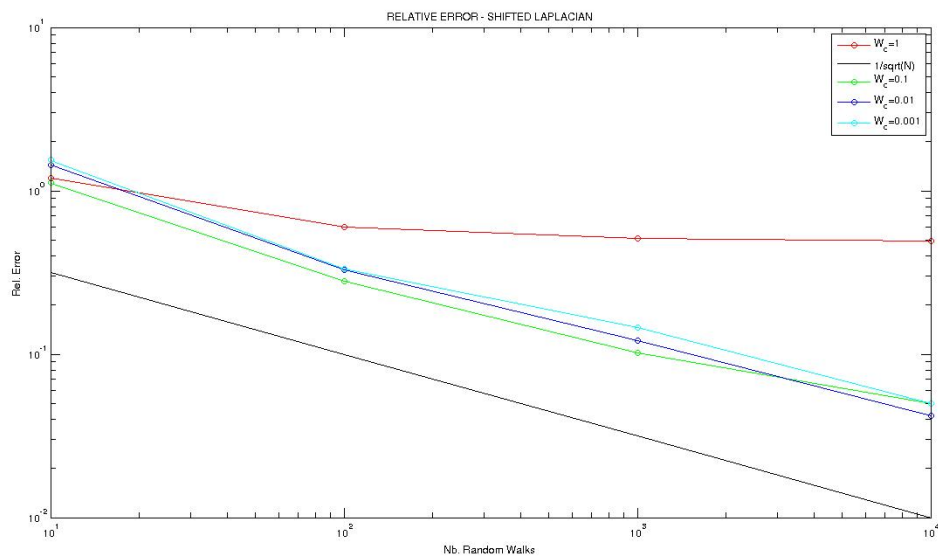


Figure 3.8: Adjoint MC: Shifted 1D Laplacian - Relative error with a weight cutoff $W_c = 1$ (red), $W_c = 0.1$ (green), $W_c = 0.01$ (blue) and $W_c = 0.001$ (cyan).

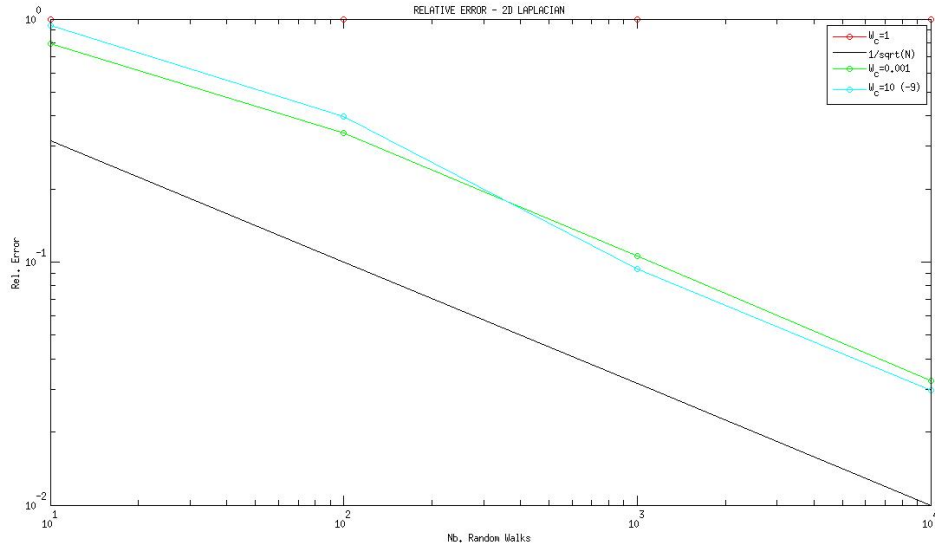


Figure 3.9: Adjoint MC: 2D Laplacian - Relative error with a weight cutoff $W_c = 1$ (red), $W_c = 0.001$ (green) and $W_c = 10^{-9}$ (cyan).

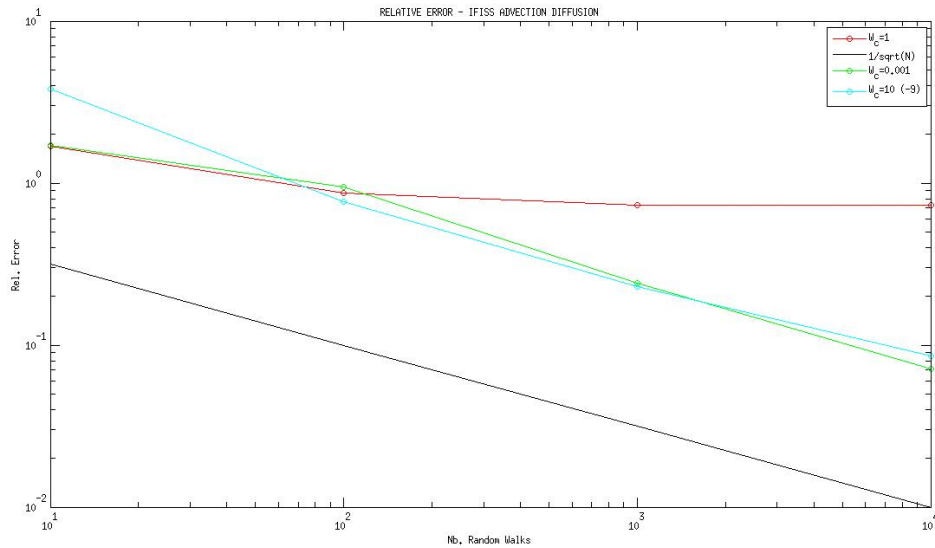


Figure 3.10: Adjoint MC: IFISS advection diffusion problem - Relative error with a weight cutoff $W_c = 1$ (red), $W_c = 0.001$ (green) and $W_c = 10^{-9}$ (cyan).

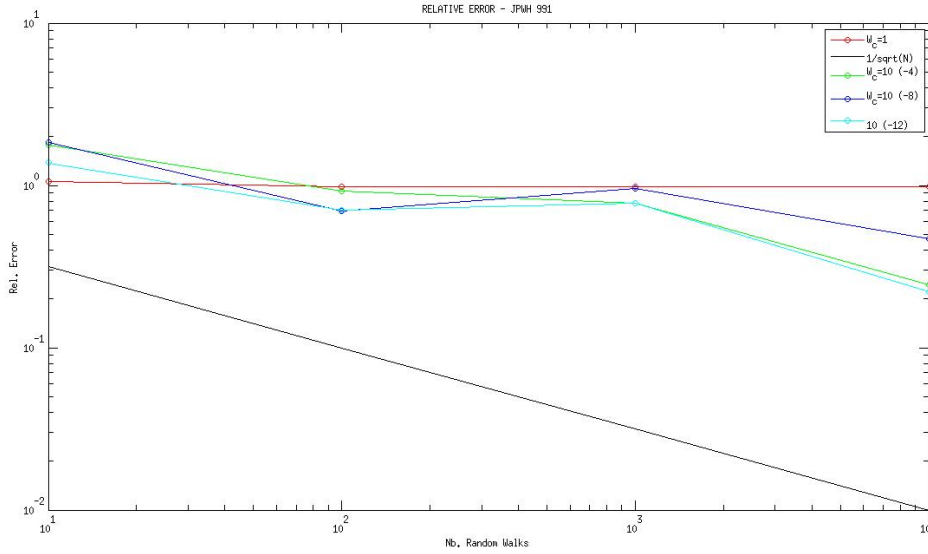


Figure 3.11: Adjoint MC: JPWH_991 - Relative error with a weight cutoff $W_c = 1$ (red), $W_c = 10^{-4}$, $W_c = 10^{-8}$ (green) and $W_c = 10^{-12}$ (cyan).

In this context, a reasonable criterion to determine the number \tilde{N}_i of random walks to be run is setting a threshold ε_1 and determine

$$\tilde{N}_i \quad s.t. \quad \frac{\sqrt{Var[\theta_i]}}{|E[\theta_i]|} < \varepsilon_1, \quad i = 1, \dots, d. \quad (3.2)$$

The dependence of $Var[\theta_i]$ and $E[\theta_i]$ on \tilde{N}_i , which seems to be absent in the previous formula, is highlighted by the fact that θ_i is estimated by fixing a finite number of histories. Therefore we are controlling the relative standard deviation requiring it not to be too large. In other words we are pursuing a statistical setting where the uncertainty factor is not dominating over the expected value. This simple adaptive approach can be applied for the estimation of each component x_i . Therefore different number of histories may be employed to compute different entries of the solution vector.

3.2.2 Relative standard deviation - Adjoint method

As we already stressed out in the previous reports, in the Adjoint method each random walk gives contributions for more than one entry. Therefore in this case the selection of the number of random walks is global, since it involves the total number of histories to estimate the entire solution vector.

As concerns the Adjoint method, the estimation of the variance for each entry is represented by Formula (2.4).

A possible adaptive selection of \tilde{N} , in this situation, is

$$\tilde{N} \quad s.t. \quad \frac{\|\sigma_{\tilde{N}}\|_1}{\|\mathbf{x}\|_1} < \varepsilon_1, \quad (3.3)$$

where σ is a vector whose entries are $\sigma_{\tilde{N},i} = Var[\theta_i]$.

3.2.3 An a posteriori variance-based adaptivity

What introduced in the two previous section can be exploited in order to build an a posteriori adaptive algorithm, capable to identify the minimal value of \tilde{N} that verifies 3.2 or 3.3 respectively.

Below there are the pseudo-codes associated with both Forward and Adjoint Monte Carlo.

Data: n, ε_1
Result: \tilde{N}, σ_i, x_i
 For each entry of the solution vector ;
 $\tilde{N}_i = n$;
 compute θ_i ;
while $\frac{\sigma_i}{|E[\theta_i]|} < \varepsilon_1$ **do**
 $\tilde{N} = \tilde{N} + n$;
 compute $E[\theta_i] = x_i$;
end
 return θ_i, x_i, σ_i ;

Algorithm 1: A posteriori adaptive Forward Monte Carlo

Data: n, ε_1
Result: $\tilde{N}, \sigma, \mathbf{x}$
 $\tilde{N} = n$;
 compute θ ;
while $\frac{\|\sigma\|}{\|E[\theta]\|} < \varepsilon_1$ **do**
 $\tilde{N} = \tilde{N} + n$;
 compute $E[\theta] = \mathbf{x}$;
end
 return $\theta, \mathbf{x}, \sigma$;

Algorithm 2: A posteriori adaptive Adjoint Monte Carlo

3.3 Numerical results

The adaptive threshold is set such that $\varepsilon_1 = 0.01$.

Forward Monte Carlo

As concerns the Forward method, the maximal admitted number of histories for each component is equal to 10 times the length of the solution vector. The number of steps per random walk is initially fixed to a constant number equal to 1000. At each adaptive iteration the number of random walks employed is increased by ten. Even if this is a very little number and it compromises the efficiency of the algorithms in terms of speed, we want to keep it small. In fact for now we want to figure out what is the minimal number of histories required for the fulfillment of the adaptive criterion. In Table 3.1 there are results for all the test cases, except for the "FS_680_1", since $\rho(H^*) = 1.2554$ implies impossibility to gain convergence.

Type of problem	Relative Error	CPU Time (s)	Nb. Histories
1d shifted Laplacian	0.0174	0.43	1324
2d Laplacian	0.132	8995	255150
JPWH_991	0.011	18058	6739640
Marshak Equation	0.0175	543	5238
ifiss			

Table 3.1: Forward Monte Carlo. Adaptive criterion.

The algorithms to detect the number of histories to run and the history's length cutoff can be combined in a unique adaptive approach. The threshold for the weight cut-off is set to 10^{-6} .

Type of problem	Relative Error	CPU Time (s)	Nb. Histories
1d shifted Laplacian	0.0145	5.26	1626
2d Laplacian	0.333	210.8	9000
ifiss advection diffusion			
JPWH_991	0.29	24.37	9910
Marshak Equation	0.017	3.54	5172

Table 3.2: Forward Monte Carlo. Adaptive cut-off and adaptive selection of # histories.

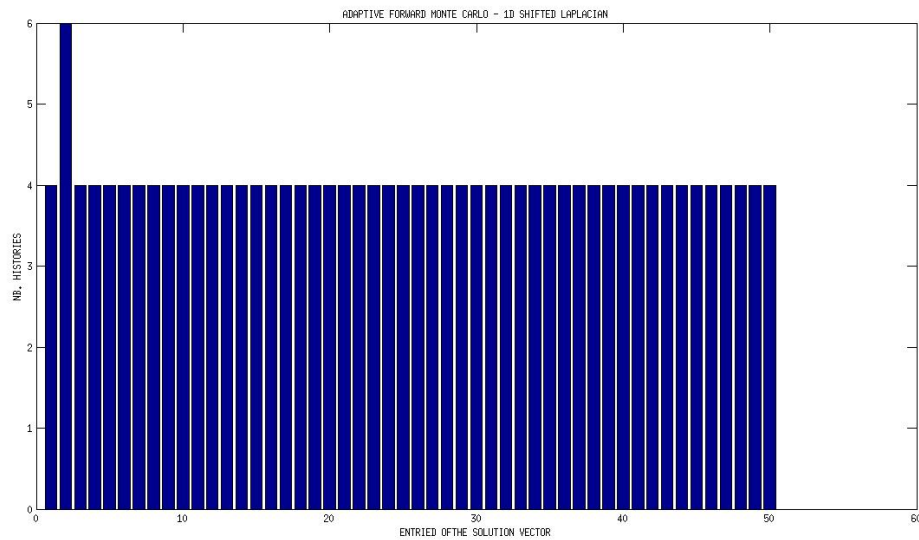


Figure 3.12: Adaptive Forward Monte Carlo - 1d shifted Laplacian - Number of random walks employed for each entry.

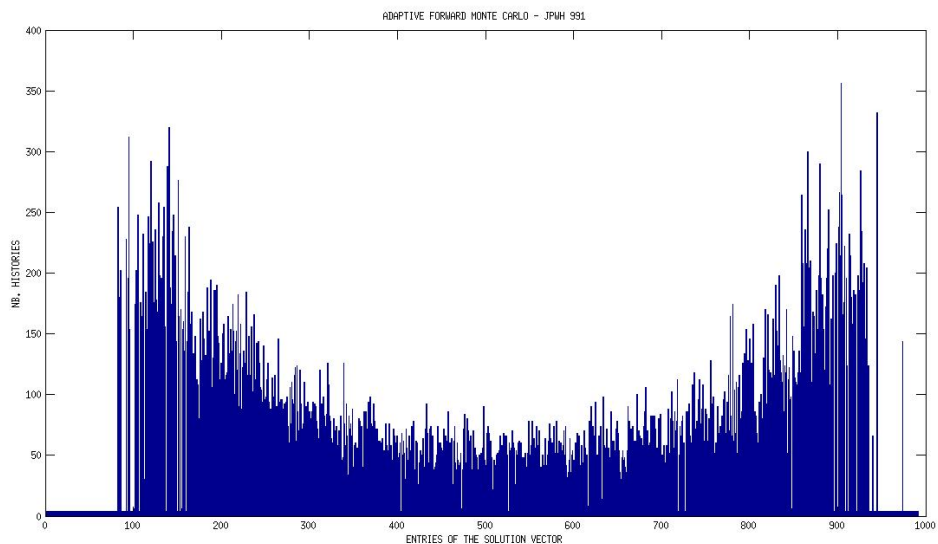


Figure 3.13: Adaptive Forward Monte Carlo - JPWH_991 - Number of random walks employed for each entry.

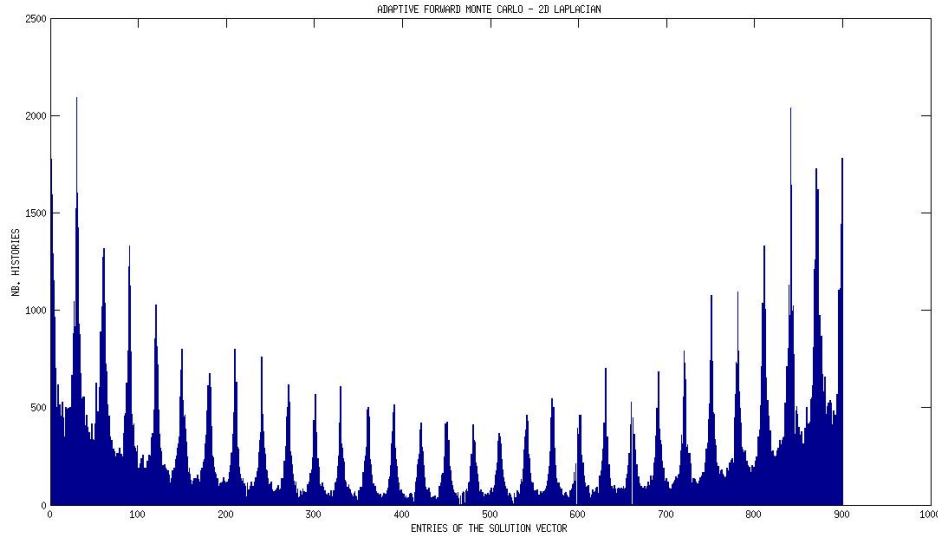


Figure 3.14: Adaptive Forward Monte Carlo - 2d Laplacian - Number of random walks employed for each entry.

Adjoint Monte Carlo

Now we get down to the Adjoint method. At each adaptive iteration the number of random walks employed is increased by two.

No restraints are put for a maximal number of random walks admitted. therefore we let the algorithm free to run until the constraint is verified.

Here below Table 3.3 shows results for the different test cases considered. As we can see, the times employed are very high. However we have to note that this is the standard Monte Carlo algorithm whose rate of convergence is the square root of the random walks computed. Thus we know it to be very slow a priori. Moreover the request of having a relative standard deviation smaller than 0.1 corresponds to a relative error of the same order. Therefore results are coherent with what expected a priori. For "JPWH_991" and "FS_680_1" the Adjoint method is not a convergent method, since $\rho(H^*) = 1.0505$ and $\rho(H^*) = 3.598$ respectively.

Type of problem	Relative Error	CPU Time (s)	Nb. Histories
1d shifted Laplacian	0.139	4.19	730
2d Laplacian	0.136	233.4	570
ifiss advection diffusion	0.282	123.7	630
Marshak equation	0.288	369	880

Table 3.3: Adjoint Monte Carlo. Adaptive criterion.

By combining adaptive cut-off for the histories and adaptive selection of the number of histories to employ, results are shown in Table 3.4. The value of the parameters has been set equal to the value used for the Forward method. By comparing the time spent we can see that the new automatic approach is highly competitive with the previous one. In fact the time spent is much less overall, without affecting the accuracy of the solution.

Type of problem	Relative Error	CPU Time (s)	Nb. Histories
1d shifted Laplacian	0.134	4.7	780
2d Laplacian	0.117	242.9	690
ifiss advection diffusion	0.343	19.5	470
Marshak equation	0.3	34.0	850

Table 3.4: Adjoint Monte Carlo. Adaptive cut-off and adaptive selection of # histories.

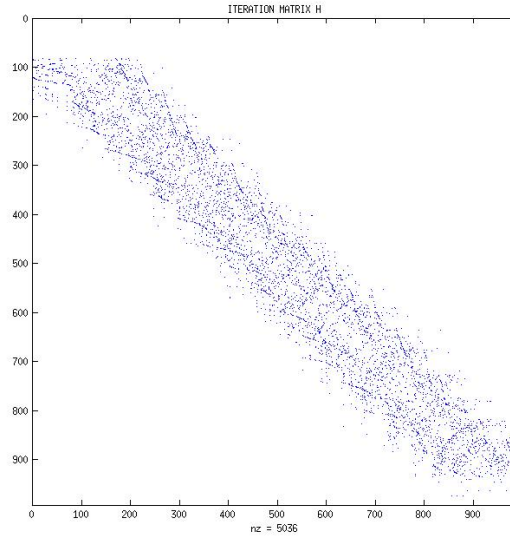


Figure 3.15: JPWH_991 - Sparsity pattern for iteration matrix H .

3.4 A stationarity check for the a posteriori adaptivity

The a posteriori approach introduced before is very naive and it does not take into account whether the number of histories is enough to reach the asymptotic trend or not. In order to do this we need to look at the value of the variance with respect to two consecutive adaptive steps. For the Forward method it means that we need to set the number of histories \tilde{N}_i such that

$$\frac{|\sigma_i^{\tilde{N}_i-1} - \sigma_i^{\tilde{N}_i}|}{|x_i|} < \varepsilon_2 \quad i = 1, \dots, n \quad (3.4)$$

where n is the size of the problem and ε_2 is a threshold set for the check of stationarity.

As concerns the Adjoint method instead we need to pursue \tilde{N} such that

$$\frac{\|\sigma^{\tilde{N}-1} - \sigma^{\tilde{N}}\|_1}{\|\mathbf{x}\|_1} < \varepsilon_2. \quad (3.5)$$

3.4 can be combined with 3.2 and 3.5 can be combined with 3.3. In this way we can build up a more robust adaptive approach that takes into account the relative magnitude of the variance and the achievement of the asymptotic behavior as well. In fact taking few histories to determine the estimation of the solution may compromise severely the accuracy of the solution itself, especially in the viewpoint of applying this approach inside of a hybrid scheme (see next chapters).

In order to verify the usefulness of using an additional check for the stationarity such as 3.4 and 3.5, we can tally the number of times the second criterion force the history to continue. For the sake of simplicity we call this phenomenon the rejection, since it represents the situation where the first criterion (3.2 or 3.3 respectively) would suggest to cut the random walk whereas the second criterion rejects this proposal.

Below we show the results obtained by combining 3.3 and 3.5 for the Forward and the Adjoint method. The test cases are the same as the ones we used in the previous section. The threshold for both the criterion is set to $\varepsilon_1 = \varepsilon_2 = 0.01$.

It is discovered that for all the test cases considered the check 3.5 forces the code to run even if 3.3 would suggest to stop the increase of histories employed. It means that the relative standard deviation is already lower than ε_1 but the slope of the standard deviation is not flat enough to catch the asymptotic behavior. Moreover, also in this case the request of having a relative standard deviation smaller than 0.01 forces the relative error to be of the same order. Thus it points out the correlation between uncertainty and accuracy associated with the solution in a stochastic linear solver.

By comparing the performance in 3.4 with the performance in 3.6 we can see that there much difference both in terms of number of random walks employed and in terms of time. Moreover the use of the stationarity check succeeds in dropping down the relative error such that it assumes the same order of magnitude as the relative standard deviation. This implies that check 3.5 is necessary to be sure that we enter the asymptotic region where the CLT actually holds. Using just check 3.3 may not guarantee that the transient regions has been overcome. Of course the combination of the two checks increases the cost of the computation. Therefore it is always necessary to ask ourselves if the increase of histories employed is really worthy.

Type of problem	Relative Error	CPU Time (s)	Rejections	Nb. Histories
1d shifted Laplacian	0.098	2.28	205	1836
2d Laplacian	0.089	164.48	0	1280
ifiss advection diffusion	0.083	630.18	2711	9800
Marshak equation	0.094	593.66	0	8018

Table 3.5: Forward Monte Carlo. Adaptive criterion with stationarity check.

Type of problem	Relative Error	CPU Time (s)	Rejections	Nb. Histories
1d shifted Laplacian	0.0077	2230	26588	266680
2d Laplacian	0.0132	22696	9323	93990
ifiss advection diffusion	0.046	20473	3623	37220
Marshak equation	0.0136	21653	39113	392070

Table 3.6: Adjoint Monte Carlo. Adaptive criterion with stationarity check.

3.5 An a priori adaptive approach on the residual

Assume θ is an estimator for a quantity x which is unknown. It holds

$$Var[\theta] = E[\theta^2] - (E[\theta])^2.$$

The theoretical variance can be estimated with a sample quantity such as the sample variance, defied as

$$\hat{\sigma} = \frac{1}{(n-1)} \sum_{i=1}^n (\theta_i - \bar{\theta})^2, \quad \bar{\theta} = \frac{1}{n} \sum_{i=1}^n \theta_i. \quad (3.6)$$

Each θ_i is a random variable associated with a single random walk. The expected value of all the random walks return the estimator we are interested in. Therefore the sample size n is the total number of histories run. Since all the random walks are independent one of the other. Therefore all the realizations of the estimator θ_i are independent as well.

This means that

$$cov(\theta_i, \theta_j) = 0, \quad i \neq j.$$

This consideration is very important for our applications. In fact it implies that the standard deviation and the statistical error both associated with the standard Monte Carlo scales with the same order.

The statistical error is defined as

$$err = \sqrt{\frac{1}{n} \left[\sum_{i=1}^n (\theta_i - x)^2 \right]}. \quad (3.7)$$

However we are assuming that θ be a unbiased estimator. This entails that statistical error and standard deviation are actually the same thing.

By combining 3.6 and 3.7 we can conclude that

$$\frac{\|\mathbf{x} - \hat{\mathbf{x}}\|}{\|\mathbf{x}\|} \sim \frac{C}{\sqrt{n}}$$

for some constant $C > 0$ and $\hat{\mathbf{x}}$ represents the standard Monte Carlo solution.

Given a linear system

$$A\mathbf{x} = \mathbf{b}, \quad A \in \mathbb{R}^{d \times d}, \quad \mathbf{x}, \mathbf{b} \in \mathbb{R}^d$$

and given $\hat{\mathbf{x}}$ the standard Monte Carlo solution, the residual is defined such as

$$\mathbf{r} = \mathbf{b} - A\hat{\mathbf{x}} = A(\mathbf{x} - \hat{\mathbf{x}}).$$

It entails that the residual has the same behavior as the error. In fact the former is a linear transformation of the latter. By looking at the behavior of the residual, we verify what expected after the reasoning accomplished. In fact the residual decreases with respect to the number of histories with a slope of $\frac{1}{\sqrt{n}}$. For a graphical representation of this look at Figures 3.16, 3.17, 3.18 and 3.19.

Hence the following relationship holds

$$\log(\|\mathbf{r}\|_2) = -\frac{1}{2} \log(n) + \log(\|\mathbf{b}\|_2) \quad (3.8)$$

regardless of the basis chosen for the logarithm. 3.8 is equivalent to

$$\log(\|\mathbf{r}\|_2) = \log\left(\frac{\|\mathbf{b}\|_2}{\sqrt{n}}\right).$$

Thus the following relationship between the residual after n histories and the right hand side holds:

$$\|\mathbf{r}\|_2 = \frac{\|\mathbf{b}\|_2}{\sqrt{n}}.$$

Since it is known that

$$\frac{\|\mathbf{x} - \hat{\mathbf{x}}\|_2}{\|\mathbf{x}\|_2} \leq K_2(A) \frac{\|\mathbf{r}\|_2}{\|\mathbf{b}\|_2} \quad (3.9)$$

By imposing

$$K_2(A) \frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} < \varepsilon, \quad 0 < \varepsilon < 1$$

we get

$$K_2(A) \frac{\|\mathbf{b}\|}{\sqrt{n}} < \varepsilon \|\mathbf{b}\|, \quad 0 < \varepsilon < 1$$

and it leads to

$$n > \frac{\left(K_2(A)\right)^2}{\varepsilon^2}. \quad (3.10)$$

3.10 represents an estimation of the number of random walks to run in order to have the guarantee that the relative error is below a certain threshold ε . Of course 3.9 may not be an effective estimation of the relative error in case of large values of the condition number. In this case the quality of 3.10 is affected as well.

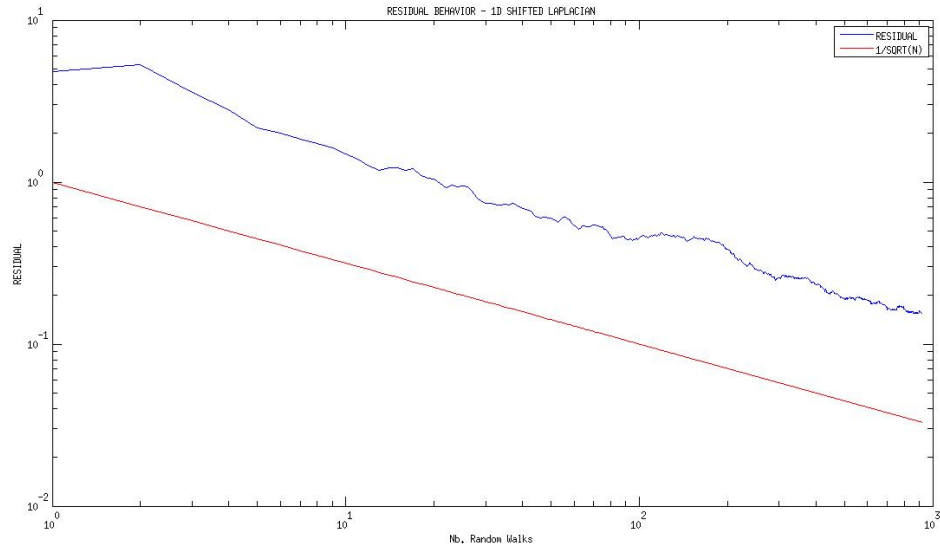


Figure 3.16: 1d shifted Laplacian - Residual behavior.

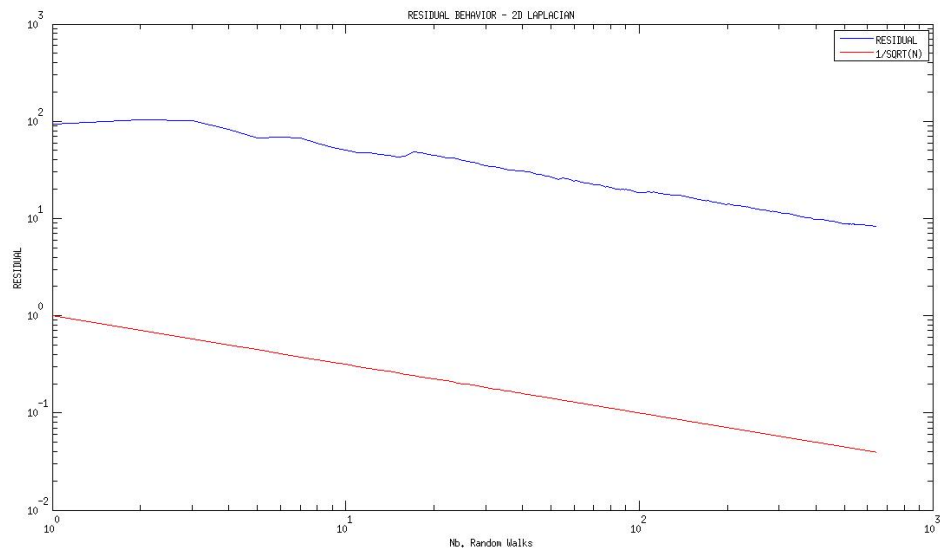


Figure 3.17: 2d Laplacian - Residual behavior.

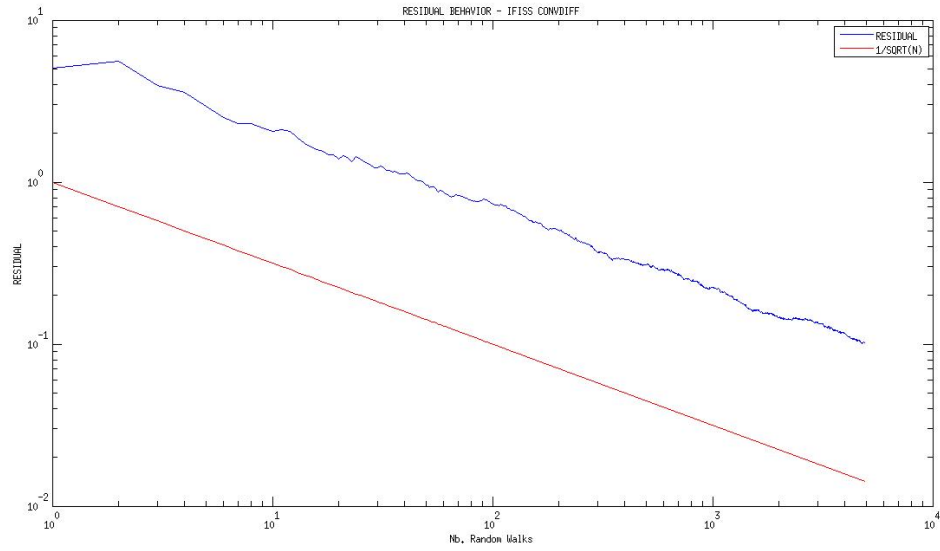


Figure 3.18: IFISS Advection Diffusion problem - Residual behavior.

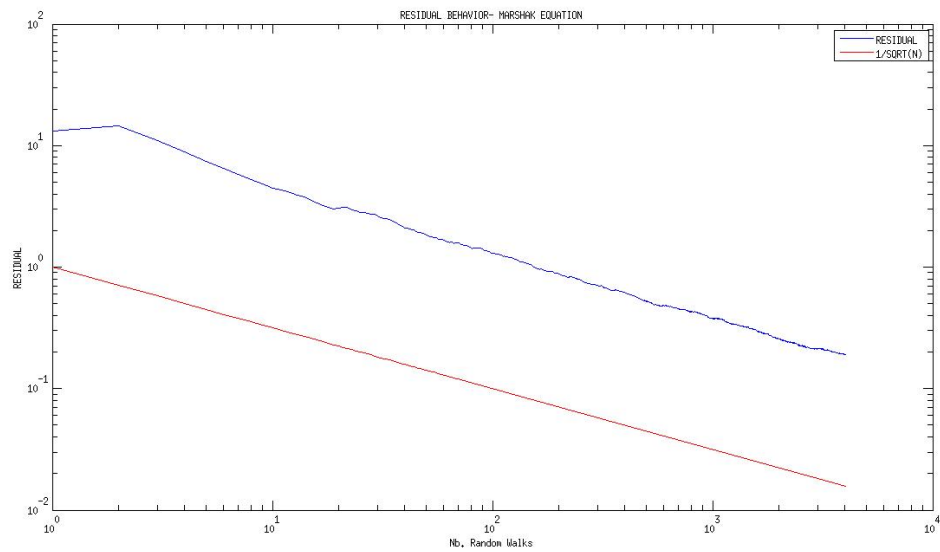


Figure 3.19: Marshak Equation - Residual behavior.

Chapter 4

Linear solvers with hybrid schemes

4.1 Sequential Monte Carlo

Even if the standard Monte Carlo, in both its Forward and Adjoint formulation, has the advantage of being embarrassingly parallelizable, it has a drawback as well. This is its slow rate of convergence, associated with the $\frac{1}{\sqrt{N}}$ behavior predicted by the CLT.

Moreover, when the spectral radius of the iteration matrix is near the unit value, Monte Carlo direct methods reach convergence after many steps for each random walk. Therefore the standard Monte Carlo paradigm is most of the time unpractical for solving linear systems associated with real applications.

In fact such this spectral configuration for matrix H is very common for linear systems derived from applicative problems (e.g. elliptic partial differential equations). Therefore the pursuit of variants for the Monte Carlo methods are necessary, in order to speed up their convergence rates. This is when iterative methods come to help.

A first step along this direction has been taken by Halton who introduced a scheme such as the following, named *Sequential Monte Carlo* ([Hal62]).

Consider a linear system like 1.1. Once it gets preconditioned and turned into a fixed point scheme, assuming that the scheme is convergent, the following algorithm may be applied to converge to the exact solution:

```
Data:  $H, \mathbf{f}, \mathbf{x}_0$ 
Result:  $x_{num}$ 
 $B = I - H;$ 
 $\mathbf{x}_{old} = \mathbf{x}_0;$ 
while not reached convergence do
     $\mathbf{r} = \mathbf{f} - B\mathbf{x}_{old};$ 
     $B\delta\mathbf{x} = \mathbf{r};$ 
     $\mathbf{x}_{new} = \mathbf{x}_{old} + \delta\mathbf{x};$ 
end
 $x_{num} = x_{new};$ 
```

Algorithm 3: Sequential Monte Carlo

This method represents an important breakthrough because it climbs over the barrier imposed by the Central Limit Theorem on a merely stochastic approach.

4.2 Monte Carlo Synthetic Acceleration

Recently Evans et al. introduced a new method: the *Monte Carlo Synthetic Acceleration (MCSA)* ([ESW13] and [EMSH14]). If we think about a fixed point formulation of the problem such as $\mathbf{x} = H\mathbf{x} + \mathbf{b}$, then the Monte Carlo Synthetic Acceleration assumes this form

Data: $H, \mathbf{b}, \mathbf{x}_0$
Result: x_{num}
 $\mathbf{x}^l = \mathbf{x}_0$;
while *not reached convergence* **do**
 $\mathbf{x}^{l+\frac{1}{2}} = H\mathbf{x}^l + \mathbf{b}$;
 $\mathbf{r}^{l+\frac{1}{2}} = \mathbf{b} - A\mathbf{x}^{l+\frac{1}{2}}$;
 $\delta\mathbf{x}^{l+\frac{1}{2}} = (I - H)^{-1}\mathbf{r}^{l+\frac{1}{2}}$;
 $\mathbf{x}^{l+1} = \mathbf{x}^{l+\frac{1}{2}} + \delta\mathbf{x}^{l+\frac{1}{2}}$;
end
 $x_{num} = x^{l+1}$;

Algorithm 4: Monte Carlo Synthetic Acceleration

The Monte Carlo method is used to compute the updating contribution $\delta\mathbf{x}^{l+\frac{1}{2}}$.

The difference between the Sequential Monte Carlo and the Monte Carlo Synthetic Acceleration consists in the first instruction of the iterative process. This is a deterministic fixed point iteration, whose purpose is to further accelerate the convergence rate.

4.3 Preconditioning techniques

4.3.1 Block Preconditioning

Consider a matrix $A \in \mathbb{R}^{n \times n}$ of the form

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1p} \\ A_{21} & A_{22} & \cdots & A_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ A_{p1} & \cdots & \cdots & A_{pp} \end{bmatrix}$$

where p is a divisor of n and such that $A_{ii} \in \mathbb{R}^{n_i \times n_i}$ are nonsingular square matrices. By defining a block diagonal matrix such as

$$D = \begin{bmatrix} A_{11} & 0_{12} & \cdots & 0_{1p} \\ 0_{21} & A_{22} & \cdots & 0_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{p1} & \cdots & \cdots & A_{pp} \end{bmatrix}$$

it is possible to compute its inverse. In particular, since the diagonal blocks do not interact one with the other, we get

$$D^{-1} = \begin{bmatrix} A_{11}^{-1} & 0_{12} & \cdots & 0_{1p} \\ 0_{21} & A_{22}^{-1} & \cdots & 0_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{p1} & \cdots & \cdots & A_{pp}^{-1} \end{bmatrix}.$$

D^{-1} can be used as a preconditioner for the original matrix A . In particular

$$D^{-1}A = \begin{bmatrix} I_{n_1 \times n_1} & A_{11}^{-1}A_{12} & \cdots & \cdots & A_{11}^{-1}A_{1p} \\ A_{22}^{-1}A_{21} & I_{n_2 \times n_2} & A_{22}^{-1}A_{23} & \cdots & A_{22}^{-1}A_{2p} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{pp}^{-1}A_{p1} & \cdots & \cdots & A_{pp}^{-1}A_{p(p-1)} & I_{n_p \times n_p} \end{bmatrix}.$$

The ultimate goal is to resort to this preconditioning technique in order to achieve the condition $\rho(D^{-1}A) < 1$. In fact this is necessary in order to reformulate 1.1 into 1.3.

Convergence of 1.3 with block diagonal preconditioning, from the deterministic point of view, is guaranteed when A is an irreducibly diagonally dominant matrix or when A is an M -matrix. In fact in situation such as these, theoretical results guarantee that $A = D - N$ is a regular splitting (see [Axe96]).

4.3.2 Approximate Inverse of a Sparse Matrix (AINV)

This stimulated our interests into other kind of preconditioners. At first we concentrated on Approximate Inverse preconditioners (see [Saa03] and [Ben02]).

The idea is to search an approximate inverse of an arbitrary sparse matrix by minimizing the Frobenius norm of the residual matrix $I - AM$:

$$F(M) = \|I - AM\|_F^2. \quad (4.1)$$

A matrix M whose value $F(M)$ is small is defined as a right-looking approximate inverse of A . In a similar way it is possible to define a left-looking approximate inverse such the the following objective function is minimized:

$$\|I - MA\|_F^2. \quad (4.2)$$

A further alternative may imply to pursuit a left-right pair L, U such that

$$\|I - LAU\|_F^2. \quad (4.3)$$

4.1 is equivalent to solve a set of decoupled least square problems:

$$F(M) = \|I - AM\|_F^2 = \sum_{j=1}^n \|e_j - Am_j\|_2^2, \quad (4.4)$$

where e_j and m_j are the j -th columns of the identity matrix and of the matrix M . An approach that may be used to solve 4.4 is to minimize separately each term of the sum in such this way:

$$f_j(m) = \|e_j - Am_j\|_2^2, \quad j = 1, 2, \dots, n.$$

Both the right-looking and the left-looking Approximate Inverse techniques cannot asses in advance whether or not the resulting M matrix is nonsingular. Therefore many times the preconditioner is built as the product of two matrices L and U that are unit lower and upper triangular

$$LAU \approx D,$$

where D is some unknown diagonal. Assuming that D is nonsingular and that $LAU = D$, then L and U are called *inverse factors* of A and we can write $A^{-1} = UD^{-1}L$. This enables to construct recursively the approximate inverse via a bordering algorithm

$$A_{k+1} = \begin{bmatrix} A_k & v_k \\ w_k & \alpha_{k+1} \end{bmatrix}$$

in which $A_n = A$.

Another technique which may be adopted to construct the approximate inverse preconditioner is the one described in [BMM96] and [BT98]. This is the actual approach we applied to the construction of the preconditioner. The specific name associated with this approach is *Factored Inverses via Orthogonalization*. The idea is to compute an approximate factorization of the form $W^T AZ = D$. W and Z are unit upper triangular matrices and D is a diagonal matrix.

In order to compute the AINV preconditioner a FORTRAN code provided by Miroslav Tuma (Academy of Sciences, Prague) is used. We report as follows the algorithm employed for the construction of the so called *Right-looking Factored AINV*. The algorithm provide the user with a dropping tolerance which can be used in order to tune the fill-in effect of the factored preconditioner.

Data: $p = q = (0, \dots, 0) \in \mathbb{R}^n$, $Z = [z_1, \dots, z_n] = I_n$, $W = [w_1, \dots, w_n] = I_n$

Result: Z, W

```

for  $k = 1, \dots, n$  do
     $p_k = w_k^T A e_k$ ,  $q_k = e_k^T A z_k$ ;
    for  $i = k + 1, \dots, n$  do
         $p_i = (w_k^T A e_i) / p_k$ ,  $q_i = (e_i^T A z_k) / q_k$ ;
        Apply a dropping rule to  $p_i, q_i$ ;
         $w_i = w_i - w_k p_i$ ,  $z_i = z_i - z_k q_i$ 
        Apply a dropping rule to  $w_{j,i}$  and  $z_{j,i}$  for  $j = 1, \dots, i$ ;
    end
;
end
;
Choose diagonal entries of  $D$  as the component of  $p$  or  $q$ ;
return  $Z, W$ ;

```

Algorithm 5: Right-looking Factored AINV

4.3.3 Incomplete LU Factorization Preconditioners (ILU)

This set of preconditioners has the goal of computing a sparse lower triangular matrix L and a sparse upper triangular matrix U such that the residual matrix $R = LU - A$ respect some constraints. One of these constraints may be the fact that some entries must be set to zero. For instance we can introduce a zero pattern set

$$P \subset \{(i, j) | i \neq j; 1 \leq i, j \leq n\}$$

and use this set in order to impose some constraints on the shape of the preconditioner built.

The way an approximate LU factorization such as this is implemented depends on the type of Gaussian elimination which is applied. For instance, the traditional three nested loops characterizing the Gaussian elimination can be introduced in different orders. This leads to different variants of the Incomplete LU Factorization algorithm (see [Saa03]). However it is possible to prove that if all the formulation are well defined, then they are all equivalent. One of these variants is the following

```

Data:  $P$ 
Result:  $A$ 
For each  $(i, j) \in P$  set  $a_{ij} = 0$ ;
for  $k = 1, \dots, n - 1$  do
    for  $i = k + 1, n$  do
        if  $(i, k) \notin P$  then
             $a_{ik} = \frac{a_{ik}}{a_{kk}}$  for  $j = k + 1, \dots, n$  and for  $(i, j) \notin P$  do
                 $a_{ij} = a_{ij} - a_{ik} * a_{kj}$ 
            end
        end
    end
end
return  $Z, W$ ;

```

Algorithm 6: General Static Pattern ILU

This kind of algorithm is guaranteed to terminate without any breakdown just for M -matrices. Assuming that the zero pattern P coincides with the zero pattern of A leads to the $ILU(0)$.


```

Data:  $P$ 
Result:  $A$ 
for  $i = 2, \dots, n$  do
  for  $k = i, i - 1$  and for  $(i, k) \in NZ(A)$  do
     $a_{ik} = \frac{a_{ik}}{a_{kk}}$  for  $j = k + 1, \dots, n$  and for  $(i, j) \in NZ(A)$  do
       $a_{ij} = a_{ij} - a_{ik}a_{kj}$ 
    end
  end
end

```

Algorithm 7: ILU(0).

More sophisticated variants of the algorithm enable to enlighten the constraint of the sparsity pattern, finding a compromise between sparsity and accuracy in approximating the inverse of A . We refer to [Saa03] and [Ben02] for further information about it.

4.4 Reordering techniques

Reordering techniques are often used in order to improve the performance of preconditioners computed through a factorization approach. The reason why they are employed is to reduce the fill-in effect, especially for ILU algorithms.

For instance, if P and Q are permutation matrices, system 1.1 may be turned into the following

$$PAQ\mathbf{y} = P\mathbf{b}, \quad \mathbf{x} = Q\mathbf{y}.$$

In general it is not possible to find a relationship between the shape of factored preconditioners associated with PAQ with respect to the preconditioners of A .

A common reordering employed for a generic matrix is the *Reverse Cuthill McKee ordering* (RCM). The purpose of this reordering is to reduce the band of nonzero entries. This way factored preconditioners such as the ones computed via ILU are expected to preserve the same bandwidth. Effectiveness of this way of proceeding in many applications is proved in [Ben02].

4.5 Numerical results

4.5.1 Illustrative examples

In this section we apply the Sequential Monte Carlo and the Monte Carlo synthetic Acceleration on the same test cases we have considered in the previous chapters. We use an adaptive approach to truncate the random walk at each numerical iteration by using a weight cut-off $W_c = 10^{-6}$. For both the Forward and the Adjoint method we use an adaptive approach also to decide how many random walks to employ at each numerical iteration. In particular as concerns the Forward method we add 10 histories at each adaptive check, whereas for the Adjoint method we compute 100 histories at each adaptive check.

The preconditioner applied on the linear systems is still a diagonal one, as for all the tests run in the previous chapters as well.

Type of problem	Rel. Residual	Rel. Error	Numerical Iter.	CPU Time (s)
1d shifted Laplacian	$1.99 \cdot 10^{-8}$	$8.64 \cdot 10^{-9}$	6	1.88
2d Laplacian	$9.54 \cdot 10^{-8}$	$4.027 \cdot 10^{-8}$	275	910
ifiss advection diffusion	$9.24 \cdot 10^{-8}$	$1.29 \cdot 10^{-6}$	73	808
JPWH 991	$8.41 \cdot 10^{-8}$	$5.21 \cdot 10^{-8}$	67	409
Marshak Equation	$1.51 \cdot 10^{-8}$	$5.68 \cdot 10^{-4}$	7	73.1

Table 4.1: Forward MCSA.

4.5.2 SP_N equations

In this subsection we focus on a set of partial differential equations which are considered a milestone in nuclear physics: the SP_N equations. These kind of equations is employed, for instance, to

Type of problem	Rel. Residual	Rel. Error	Numerical Iter.	CPU Time (s)
1d shifted Laplacian	$1.99 \cdot 10^{-8}$	$8.64 \cdot 10^{-9}$	6	1.88
2d Laplacian	$9.54 \cdot 10^{-8}$	$4.027 \cdot 10^{-8}$	275	910
ifiss advection diffusion	$9.24 \cdot 10^{-8}$	$1.29 \cdot 10^{-6}$	73	808
Marshak Equation	$1.51 \cdot 10^{-8}$	$5.68 \cdot 10^{-4}$	7	73.1

Table 4.2: Adjoint MCSA.

synthesize phenomena such as the power distribution cross a nuclear reactor core (see [EMSH14]). Physical events of this type are extremely complicated and practical devices able to analyze them in their entire complexity are not at our hand currently. Indeed the starting equation, used as a reference to represent the evolution of phenomena such as these, is the Boltzmann equation. In a multidimensional domain the terms considered inside of the equation are difficult to be treated and state-of-the art modelings are not capable of handling the complexity of the original model. Therefore successive simplifications are introduced in order to formulate a treatable and practical representation of the starting problem.

For example, let us consider a steady-state, multi-group, one-dimensional, eigenvalue-form of Boltzmann:

$$\begin{aligned} \mu \frac{\partial \psi^g(x, \mu)}{\partial x} + \sigma^g(x) \psi^g(x, \mu) = \sum_{g'=1}^{N_g} \int_{4\pi} \sigma_s^{gg'}(x, \hat{\Omega} \cdot \hat{\Omega}') \psi^{g'}(x, \Omega') d\Omega' + \\ + \frac{1}{k} \sum_{g'=1}^{N_g} \frac{\chi^g}{4\pi} \int_{4\pi} \nu \sigma_f^{g'}(x) \psi^{g'}(x, \Omega') d\Omega'. \end{aligned} \quad (4.5)$$

The meaning of the quantities appearing in 4.5 is the following:

- N_g = total number of levels of energy used to classify neutrons
- $\psi^g(x, \mu)$ = angular flux for group g
- σ^g = total interaction cross section
- $\sigma_s^{gg'}(x, \hat{\Omega} \cdot \hat{\Omega}')$ = scattering cross section from group $g' \rightarrow g$
- χ^g = resulting fission spectrum in group g
- k = ratio of neutron populations in subsequent fission generations.

By resorting to a spectral decomposition of the angular flux ψ and of the interaction σ by Legendre polynomials we get

$$\psi^g(\mu) = \sum_{i=0}^{\infty} \frac{2i+1}{4\pi} \varphi_i^g p_i(x)$$

and

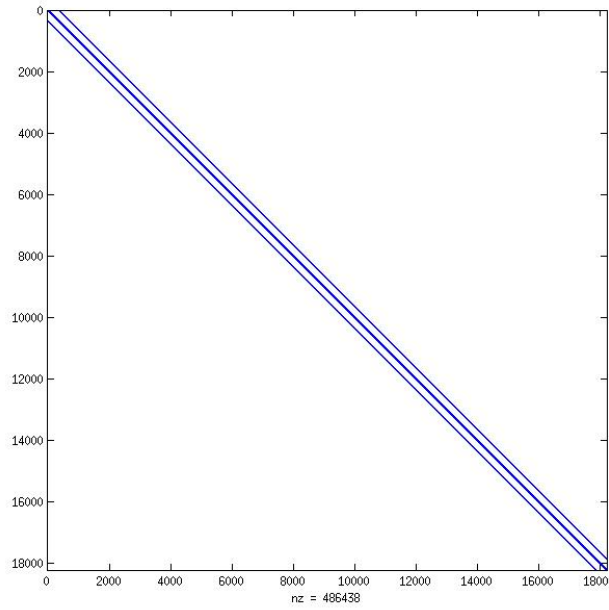
$$\sigma_s(\mu_0) = \sum_{j=0}^{\infty} \frac{2j+1}{4\pi} \sigma_{sj} p_j(x).$$

The truncation of the Fourier series to the N -th term we get to the formulation of the so called P_N equations:

$$\frac{\partial}{\partial x} \left[\frac{i}{2i+1} \varphi_{i-1}^g + \frac{i+1}{2i+1} \varphi_{i+1}^g \right] + \sum_{g'=1}^{N_g} (\sigma^g \delta_{gg'} - \sigma_{sn}^{gg'}) \varphi_i^{g'} = \frac{1}{k} \sum_{g'=1}^{N_g} \chi^g \nu \sigma_f^{g'} \varphi_i^{g'} \delta_{i0}, \quad i = 0, 1, 2, \dots, N.$$

Considering just odd sets of P_N equations and by removing lower gradient terms from each equation we get to the formulation of simplified P_N equations (SP_N):

$$-\nabla \cdot \mathbb{D}_i \nabla \mathbb{U}_i + \sum_{j=0}^{\frac{N+1}{2}} \mathbb{A}_{ij} \mathbb{U}_j = \frac{1}{k} \sum_{j=1}^{\frac{N+1}{2}} \mathbb{F}_{ij} \mathbb{U}_j, \quad i = 1, \dots, \frac{N+1}{2} \quad (4.6)$$

Figure 4.1: Sparsity pattern of the SP_1 equation.

\mathbb{U}_j is a linear combination of moments.

In the problems we focused on, a Finite Volume discretization has been applied in order to build the linear system to solve.

We consider two examples of SP_1 equations, an example of SP_3 and one of SP_5 . In particular for the SP_{11} equations the matrices have respectively size 18207×18207 and 19941×19941 . Their sparsity pattern is shown in Figures 4.1 and 4.2. The SP_3 matrix is a 36414×36414 matrix while the SP_5 one is a 56621×56621 .

The goal is to detect a suitable preconditioner to apply to the linear system in order to make the MC solver converge. As it will be further explained in the following chapter, having a diagonally dominant structure might be helpful but actually this is not our case. In fact we do not succeed in gaining convergence of the Monte Carlo linear solver with a diagonal preconditioning for these problems.

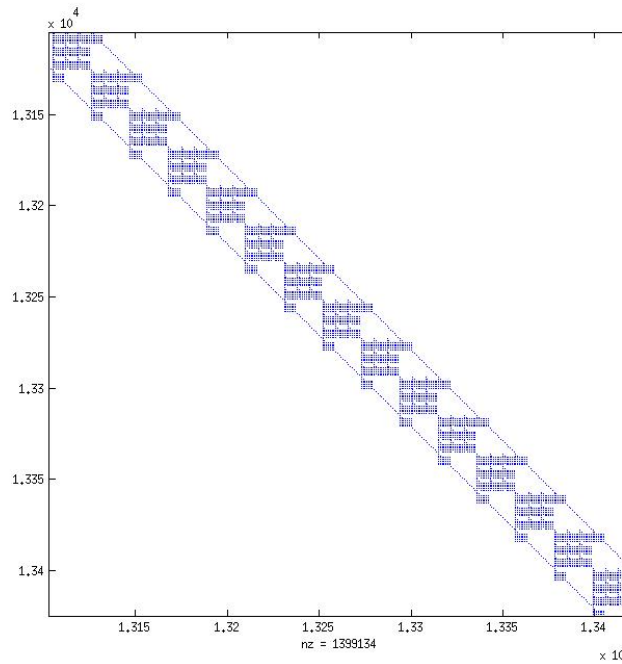
Therefore a block diagonal preconditioner is initially tested, by varying the size of the blocks in a reasonable range of values. In fact it is very important to preserve the sparsity pattern of the ultimate linear system. It turns out that for the two cases of SP_1 equations we have to apply block diagonal preconditioners of sizes respectively 63×63 and 69×69 in order to get the conditions stated in [HMY13] verified.

The effect of the application of the preconditioner in terms of fill-in effect are represented in Table 4.3. For the matrix (b) the preconditioner is quite efficient since it does not change much the number of nonzero entries in the matrix. The same does not hold for matrix (a), since in this case the preconditioner increases the number of nonzero entries by a factor of six.

Kind of matrix	size	nnz	Prec. block size	nnz after prec
SP_1 (a)	18207	486438	63	2644523
SP_1 (b)	19941	998631	69	1774786

Table 4.3: SP_1 matrices. Analysis of sparsity.

Numerical results are shown in Table 4.4. In terms of number of numerical iterations employed the results are satisfactory. As concerns the times, the slowness of the method is an index of the fact that most of the work is executed by the stochastic part of the algorithm. Therefore most of the computation is relegated into the random walks which in this case are all run in a serial mode.

Figure 4.2: Zoom on the sparsity pattern of the SP_1 equation.

Therefore, even if the times themselves are not satisfactory, they are promising in the viewpoint of a future parallelization of the method.

matrix	$\rho(H)$	$\rho(H^*)$	relative err.	# iterations	CPU time (s)
SP_1 (a)	0.9779	0.9758	$9.97 \cdot 10^{-6}$	340	17035 ($\approx 5h$)
SP_1 (b)	0.9798	0.9439	$3.89 \cdot 10^{-5}$	209	11204 ($\approx 3h$)

Table 4.4: SP_1 matrices. Block diagonal preconditioning. Numerical results

The block diagonal preconditioning is not successful in general on SP_N matrices. In fact by adopting the same approach on SP_3 and SP_5 matrices it was not possible to find a configuration such that the MC solvers would converge. Even by varying the size of the blocks, it is easy to have $\rho(H) < 1$ but the same does not hold for H^* . Both left and right preconditioning have been tried out as well.

We have computed the AINV preconditioner on all the SP_N matrices at hand to pursue a configuration suitable for us. In order to accomplish this, we tuned the drop tolerance that controls the fill-in phenomenon of the preconditioner. Once this done, we have to choose the position where to apply the preconditioner, on the left rather than on the right.

In fact in the deterministic environment this would not make any substantial difference, since the resulting iteration matrix would have the same spectral radius regardless of the position where the preconditioning is applied. However the same does not hold in the definition of the Monte Carlo linear solver. In fact the matrix H^* is defined entry-wise in terms of H . Therefore the position where the preconditioner is applied affects the values attained by the entries of H^* , affecting its spectral radius as well.

In conclusion the position of the preconditioning may determine the convergence of the stochastic scheme.

An illustrative example in this case is represented indeed by the AINV preconditioning over the two SP_1 matrices at hand.

For the $SP_1(a)$ we compute a factored AINV preconditioner with a drop tolerance of $\varepsilon = 0.001$ for both the factors. In Table 4.5 we report the value of the spectral radii of H^* for $SP_1(a)$ and $SP_1(b)$ when the preconditioning is applied on the left and on the right respectively.

Initial matrix	$\rho(H)$	$Leftprec - \rho(H^*)$	$Rightprec - \rho(H^*)$
SP_1 (a)	0.722	1.240	0.920
SP_1 (b)	0.644	1.185	0.998

Table 4.5: Spectral radii of H and H^* for different AINV preconditioning positions.

Computing the entity of the fill-in for the right preconditioning we get

$$\text{ratio} = \frac{nnz(H)}{nnz(A)} = 22.07,$$

where $H = I - AP^{-1}$ and A is the matrix associated with the $SP_1(a)$ problem.

Therefore the fill-in effect compromises the sparsity pattern.

As regards the $SP_2(b)$ matrix an AINV preconditioner with a drop tolerance of $\varepsilon = 0.0009$ for both the factors has been computed. Checking the magnitude of the fill-in phenomenon for the right preconditioning we get

$$\text{ratio} = \frac{nnz(H)}{nnz(A)} = 14.94.$$

In this case we have a massive fill-in as well.

In Table 4.6 we report the numerical results associated with the two aforementioned cases.

matrix	relative err.	# iterations	CPU time (s)
SP_1 (a)	$6.277 \cdot 10^{-7}$	33	900 ($\approx 15\text{min}$)
SP_1 (b)	$9.885 \cdot 10^{-7}$	21	650 ($\approx 10\text{min}$)

Table 4.6: Numerical results for different AINV preconditioning positions.

Both in terms of numerical iterations and time employed the AINV preconditioning is competitive with respect to the Block Jacobi. However the sparsity of the preconditioned system is partially worsened by the low value of the drop tolerance imposed. Either by increasing the value of ε or by applying a filtering on the computed preconditioner, the condition $\rho(H^*)$ is lost as concerns the SP_1 matrices available.

We still aim at applying the AINV preconditioner on other sets of problems, where the structure of the matrix may facilitate the AINV algorithms in providing us with a preconditioner which guarantees convergence for the Monte Carlo linear solvers.

The ILU preconditioner has been computed as well. However even for ILU(0) the sparsity pattern of the preconditioner is inevitably lost. Therefore it was not worth keeping on studying this kind of preconditioners for the problems considered in this section.

As already said about the AINV preconditioners, we preserve our interest for other categories of problems for which the computation of an ILU preconditioner may actually be profitable from the convergence viewpoint.

Chapter 5

Applicable sets for MC solvers

As we have already seen in Chapter 2, sufficient conditions for the convergence of the Monte Carlo linear are very unlikely to be applicable because of the restricted cases of interest (see [HMY13]). Moreover the necessary and sufficient condition requires the computation of $\rho(H)$ and $\rho(H^*)$. The former may be always guaranteed, once a proper preconditioner is picked (e.g. Approximate Inverse). The latter is more problematic to be verified. Therefore in most of the cases it should be checked every time.

The computational burden associated with the spectral radius of H^* has the same complexity as a deterministic algorithm employed to compute the solution to the linear system 1.1 we are interested in.

In order to bypass this task, it is necessary to look for set of matrices for which the Monte Carlo linear solver always converges, maybe with the use of an ad-hoc preconditioner.

5.1 Strictly diagonally dominant matrices

One of the types of matrices for which the MC solver always converges is represented by strictly diagonally dominant matrices.

Definition 1 A matrix $A \in \mathbb{R}^{n \times n}$ is strictly diagonally dominant (s.d.d) by rows if

$$|a_{ij}| > \sum_{\substack{i=1 \\ i \neq j}}^{i=n} |a_{ij}| \quad (5.1)$$

Definition 2 A matrix $A \in \mathbb{R}^{n \times n}$ is strictly diagonally dominant (s.d.d) by columns if

$$|a_{ij}| > \sum_{\substack{j=1 \\ j \neq i}}^{j=n} |a_{ij}| \quad (5.2)$$

When 5.1 holds we can resort to a left diagonal preconditioning and we get an iteration matrix $H = I - D^{-1}A$ such that $\|H\|_\infty < 1$.

Introducing a MAO transition probability for the Forward Method we get

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

therefore the entries of H^* are defined as follows

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

Therefore

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

This automatically implies that $\rho(H^*) \leq \|H^*\|_\infty < 1$. Thus the Forward Monte Carlo converges. However nothing a priori can be stated about the convergence of the Adjoint method.

Instead if 5.2 holds we can resort to a right diagonal preconditioning and we get an iteration matrix $H = I - AD^{-1}$ such that $\|H\|_1 < 1$. In this case, by following a similar reasoning to the one made before, we conclude that the Adjoint Method converges because $\|H^*\|_1 < 1$, whereas nothing can be said in advance as concerns the Forward method.

5.1.1 Numerical results

In this section of numerical experiments we focus on a set of matrices obtained by a diagonal shift of the matrices associated with SP_1 , SP_3 and SP_5 equations. The shift applied is such that

$$(A + sD), \quad s \in \mathbb{R}^+, \quad D = \text{diag}(A).$$

All the matrices have been turned into strictly diagonally dominant by columns. Therefore the Adjoint Monte Carlo is expected to converge in all the cases.

In Table 5.1 we report the values of the spectra of H and H^* . For the sake of completeness we have computed the values for both the Forward and the Adjoint Monte Carlo but we will focus on the latter for the computations.

The setting adopted for the parameters of the Adjoint MCSA algorithm is as follows:

- residual relative tolerance: $\varepsilon_1 = 10^{-7}$
- almost optimal transition probability
- maximal # steps per history: 10
- statistical error-based adaptive parameter: $\varepsilon_2 = 0.5$
- granularity of the adaptive approach: $n_{histories} = 1000$

Numerical results are shown in 5.2.

Initial matrix	s	$\rho(H)$	$Forward - \rho(H^*)$	$Adjoint - \rho(H^*)$
SP_1 (a)	0.3	0.7597	0.7441	0.6983
SP_1 (b)	0.4	0.7046	1.1448	0.5680
SP_3	0.9	0.5869	0.4426	0.3727
SP_5	1.6	0.5477	0.3790	0.3431

Table 5.1: S.d.d. diagonally shifted SP_N . Spectra of H and H^* - Forward and Adjoint method

matrix	nnz	s	relative err.	# iterations	CPU time (s)
SP_1 (a)	486438	0.3	$6.277 \cdot 10^{-7}$	36	2271 ($\approx 40\text{min}$)
SP_1 (b)	998631	0.4	$9.885 \cdot 10^{-7}$	21	1172 ($\approx 15\text{min}$)
SP_3	846549	0.9	$5.919 \cdot 10^{-7}$	18	3663 ($\approx 40\text{min}$)
SP_5	1399134	1.6	$4.031 \cdot 10^{-7}$	19	6491 ($\approx 110\text{min}$)

Table 5.2: S.d.d. diagonally shifted SP_N . Adjoint MCSA. Numerical results

The condition of strictly diagonally dominance is a sufficient condition to gain the converge of the algorithm. Nevertheless it is not necessary at all. Indeed, by reducing the entity of the diagonal shift, we lose the s.d.d. property but the stochastic linear solver provides an accurate solution as well. In Table 5.3 and 5.4 we have respectively the spectral radii and the numerical results. The spectral radii have increased with respect to the previous set of cases. In fact the property of diagonally dominance sped up the computations. Once this property is missing, the convergence of the algorithm may be preserved but the number of numerical iterations and the total time employed upsurge.

Initial matrix	s	$\rho(H)$	$Forward - \rho(H^*)$	$Adjoint - \rho(H^*)$
SP_1 (a)	0.2	0.8230	0.8733	0.8195
SP_1 (b)	0.2	0.8220	1.5582	0.7731
SP_3	0.3	0.8126	0.9459	0.7961
SP_5	0.7	0.8376	0.8865	0.8026

Table 5.3: Diagonally shifted SP_N . Spectra of H and H^* - Forward and Adjoint method

matrix	s	relative err.	# iterations	CPU time (s)
SP_1 (a)	0.2	$6.394 \cdot 10^{-7}$	48	3438 ($\approx 1h$)
SP_1 (b)	0.2	$2.59 \cdot 10^{-6}$	36	1581 ($\approx 30min$)
SP_3	0.3	$5.35 \cdot 10^{-7}$	45	9715 ($\approx 150min$)
SP_5	0.7	$4.21 \cdot 10^{-7}$	70	22696 ($\approx 6h$)

Table 5.4: Diagonally shifted SP_N . Adjoint MCSA. Numerical results

5.2 Generalized diagonally dominant matrices (GDDM)

Another set of matrices for which it is possible to guarantee convergence of the Monte Carlo linear solvers is represented by *generalized diagonally dominant matrices*.

Definition 3 A square matrix $A \in \mathbb{R}^{n \times n}$ is said to be *generalized diagonally dominant* if

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

for some positive vector $\mathbf{x} = (x_1, \dots, x_n)^T$.

A proper subset of the generalized diagonally dominant matrices is represented by M -matrices (see [Axe96]).

There are many definitions of M -matrices that may be found on books and all of them are equivalent. The one presented in [Saa03] states the following

Definition 4 A matrix $A \in \mathbb{R}^{n \times n}$ is said to be an M -matrix if it satisfies the following four properties:

- $a_{i,i} > 0$ for $i = 1, \dots, n$
- $a_{i,j} \leq 0$ for $i \neq j$, $i, j = 1, \dots, n$
- A is nonsingular
- $A^{-1} \geq 0$

In order to motivate the assertion that M -matrices are generalized diagonally dominant, we will show as follows some theoretical results.

For the sake of utility let us introduce also the concept of *generalized positive type*.

Definition 5 (Bramble and Hubbard, 1964) A real matrix $A \in \mathbb{R}^{n \times n}$, with $a_{i,j} \leq 0$ for $i \neq j$, is said to be of *generalized positive type* if there exists a real vector $\mathbf{x} > 0$, $\mathbf{x} \in \mathbf{R}^n$, such that:

- $A\mathbf{x} \geq 0$ and $J(A) = i; (A\mathbf{x})_i > 0 \neq \emptyset$, that is, the index set $J(A)$ is nonempty.
- If $i \notin J(A)$, there exists indexes $i_1, i_2, \dots, i_r \in \{1, 2, \dots, n\}$, such that $a_{i_k, i_{k+1}} \neq 0$, $k = 1, 2, \dots, r-1$, $i_1 = i$, and $i_r \in J(A)$. Equivalently, there exists a path in the directed graph of A from any index $i \notin J(A)$ to any index in $J(A)$.

In [Axe96] the following theorem, attributed to Beauwens and Varga is stated

Theorem 6 Let $A \in \mathbb{R}^{n \times n}$, with $a_{i,j} \leq 0$ for $i \neq j$. Then the following are equivalent:

- A is of generalized positive type
- There exists an $\mathbf{x} > 0$ and a permutation matrix P , such that $PA\mathbf{x} \geq 0$ and

$$\sum_{j=1}^i b_{i,j} x_j > 0, \quad i = 1, 2, \dots, n, \quad \text{where} \quad b_{i,j} = (PA)_{i,j}$$

- A is generalized strictly diagonally dominant, and $a_{ii} \geq 0$
- A is a M -matrix.

Very important properties of M -matrices are that principal matrices and direct sums of M -matrices are still M -matrices (see [HJ94] and [Axe96]). Even more M -matrices are positive stable, which means that all the eigenvalues have nonnegative real part. These statements have been proved by Ostrowski in 1937.

The following theorem instead is useful in order to build a convergent splitting.

Theorem 7 *Let $A \in \mathbb{R}^{n \times n}$ be a M -matrix that is partitioned in block matrix form. Then*

- The matrices $A_{i,i}$ on the diagonal of A are M -matrices
- The block lower and upper triangular part of A are M -matrices

It implies that Jacobi, Block Jacobi and Gauss Seidel are convergent splitting for an M -matrix. However, all this useful properties are not enough to ensure the convergence of the Monte Carlo linear solver. In fact none of these conditions is sufficient to guarantee $\rho(H^*) < 1$.

In order to restore convergence for a generalized diagonally dominant matrix we used a result shown in [Li02]. In this paper the author present an algorithm to transform a generalized diagonally dominant matrix with nonzero diagonal entries into strictly diagonally dominant. The algorithm works for a generic complex matrix $A \in \mathbb{C}^{n \times n}$.

Here below we report the algorithm at hand

For a given complex matrix A , $a_{ii} \neq 0$, $i = 1, \dots, n$;

1. Compute $S_i = \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|$, $i = 1, 2, \dots, n$
2. Set $t = 0$. For $i = 1, 2, \dots, n$, if $|a_{ii}| > S_i$, then set $t = t + 1$
3. If $t = 0$, then print "A is not a GDDM": END
4. If $t = n$, then print "A is a GDDM": END
5. **for** $i=1, n$ **do**

$d_i = \frac{S_i + \varepsilon}{ a_{ii} + \varepsilon} \quad \varepsilon > 0, \quad j = 1, 2, \dots, n;$	
$a_{ji} = a_{ji} \cdot d_i$	
end	
6. Go to step 1.

Algorithm 8: Algorithm to turn a GDDM matrix into a sdd by rows.

This approach turns a generalized diagonally dominant matrix into a strictly diagonally dominant matrix by rows. By substituting $S_i = \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|$ at step 1 with $S_i = \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ji}|$ 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 GDDM matrix into a s.d.d. by columns.

Once we have applied this transformation to the matrix at hand into a s.d.d., we can use the Monte Carlo linear solver which is ensured to converge.

5.3 "Block-diagonally dominant"-like matrices

In Chapter 4 we have already discussed the Block diagonal preconditioning. In this section we wonder in which situations a preconditioning such as this guarantees to come up with a convergent Monte Carlo linear solver. By mimicking the computations we already showed previously, the iteration matrix $H \in \mathbb{R}^{n \times n}$ resulting from a block diagonal preconditioning assumes the form

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

5.3.1 Forward method

By assuming that all n_i have the same value we may define

$$m = \text{size of a block} = \frac{n_i}{p}.$$

The MAO transition probability matrix assumes the following form:

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

Consequently, the H^* matrix is defined such that

$$H_{i,j}^* = |H_{i,j}| \left(\sum_{k=1}^n |H_{i,k}| \right) = \left| \left(A_{\lfloor \frac{i}{m} \rfloor \lfloor \frac{j}{m} \rfloor}^{-1} A_{\lfloor \frac{i}{m} \rfloor \lfloor \frac{j}{m} \rfloor} \right)_{(i \% m)(j \% m)} \right| \sum_{\substack{k=1 \\ k \neq i}}^n \left| \left(A_{\lfloor \frac{i}{m} \rfloor \lfloor \frac{k}{m} \rfloor}^{-1} A_{\lfloor \frac{i}{m} \rfloor \lfloor \frac{k}{m} \rfloor} \right)_{(i \% m)(k \% m)} \right|$$

By computing the sum over a generic row of H^* we get:

$$\sum_{j=1}^n |H_{i,j}^*| = \sum_{j=1}^n H_{i,j}^* = \left(\sum_{\substack{j=1 \\ j \neq i}}^n \left| \left(A_{\lfloor \frac{i}{m} \rfloor \lfloor \frac{j}{m} \rfloor}^{-1} A_{\lfloor \frac{i}{m} \rfloor \lfloor \frac{j}{m} \rfloor} \right)_{(i \% m)(j \% m)} \right| \right)^2$$

if we focus on the norm $\|H^*\|_\infty$, then the following equivalence condition holds:

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

A sufficient condition for this to happen is

$$\sum_{\substack{j=1 \\ j \neq i}}^p \|A_{ii}^{-1} A_{ij}\|_\infty < 1. \quad (5.3)$$

By defining a matrix $\tilde{H} \in \mathbb{R}^{p \times p}$ such that

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

We can use the matrix \tilde{H} matrix just defined in order to introduce a sufficient condition for the convergence of the Forward Monte Carlo method with a Block Diagonal preconditioning.

$$\|\tilde{H}\|_\infty < 1 \Rightarrow \|H^*\|_\infty < 1. \quad (5.4)$$

5.3.2 Adjoint method

Analogously to the Forward method, if we define

$$(H^*)_{i,j}^T = |H_{i,j}^T| \left(\sum_{k=1}^n |H_{ik}^T| \right)$$

we can formulate a sufficient condition for the convergence of the Adjoint Monte Carlo method by introducing a matrix \tilde{H} which in this case is such that

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

The sufficient condition assume the form

$$\|\tilde{H}\|_1 < 1 \Rightarrow \|H^*\|_1 < 1. \quad (5.5)$$

Bibliography

- [AAD⁺05] V. Alessandrov, E. Atanassov, I. Dimov, S. Branford, A. Thandavan, and C. Weihrauch. Parallel hybrid Monte Carlo algorithms for matrix computations problems. *Lecture Notes in Computer Science*, 3516:752–759, 2005.
- [Axe96] O. Axelsson. *Iterative Solution Methods*. Cambridge University Press, 1996.
- [Ben02] M. Benzi. Preconditioning Techniques for Large Linear Systems: A Survey. *Journal of Computational Physics*, 182:418–477, 2002.
- [BMM96] M. Benzi, C.D. Meyer, and Tuma M. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal on Scientific Computing*, 17:1135–1149, 1996.
- [BT98] M. Benzi and M. Tuma. A sparse approximate inverse preconditioner for nonsymmetric linear systems. *SIAM Journal on Scientific Computing*, 19:968–994, 1998.
- [DA98] I.T. Dimov and V.N. Alexandrov. A new highly convergent Monte Carlo method for matrix computations. *Mathematics and Computers in Simulation*, (47):165–181, 1998.
- [DAK01] I. Dimov, V. Alexandrov, and A. Karaivanova. Parallel resolvent Monte Carlo algorithms for linear algebra problems. *Mathematics and Computers in Simulation*, 55(55):25–35, 2001.
- [EMSH14] 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(November 2013):338–358, 2014.
- [ESW13] T.M. Evans, S.R. Slattery, 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.
- [Hal62] J.H. Halton. Sequential Monte Carlo. *Mathematical Proceedings of the Cambridge Philosophical Society*, 58(1):57–58, 1962.
- [Hal94] J.H. Halton. Sequential Monte Carlo techniques for the solution of linear systems. *Journal of Scientific Computing*, 9(2):213–257, 1994.
- [HJ94] R. A. Horn and C. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1994.
- [HMY13] J. Hao, M. Mascagni, and L. Yaohang. Convergence analysis of Markov chain Monte Carlo linear solvers using Ulam-von Neumann algorithm. *SIAM Journal on Numerical Analysis*, 51(4):2107–2122, 2013.
- [Li02] L. Li. On the iterative criterion for generalized diagonally dominant matrices. *SIAM Journal in Matrix Analysis and Applications*, 24(1):17–24, 2002.
- [Saa03] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2003.
- [Sla13] S. Slattery. *Parallel Monte Carlo Synthetic Acceleration Methods For Discrete Transport Problems*. PhD thesis, University of Wisconsin-Madison, 2013.