MONTE CARLO METHODS TO SOLVE SPARSE LINEAR SYSTEMS

Report 1

Massimiliano Lupo Pasini

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 Analaysis. 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 facilitites (which stands for computing machines with 10^6 cores), 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], [AAB$^+$05],[ESW13] and [EMSH14]).

In particular, the main issue is a comparison between two different approaches, both aiming to solve 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.

After a brief introduction to the numerical scheme of the aforementioned methods, different test cases are presented with results. The section dealing with test cases is split up into two parts: in the first one Monte Carlo method is used as a solver, in the second one it is used as a preconditioner inside of a Richardson scheme.

Chapter 1

Mathematical methods

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

$$A\mathbf{x} = \mathbf{b}, \tag{1.1}$$

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

We know that system (1.1) can be reinterpreted as a fixed point problem

$$\mathbf{x} = H\mathbf{x} + \mathbf{b}. \tag{1.2}$$

A possible choice for H may be $H = I - A$, but we will discuss later some different choices that can be done.

The solution to (1.2) can be written in terms of a power series of H (Neumann-Ulam series)

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

The power series is guaranteed to converge if the spectral radius $\rho(H) < 1$.

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

$$x_i = \sum_{l=0}^{\infty} \sum_{k_1}^{n} \sum_{k_2}^{n} \cdots \sum_{k_l=1}^{n} H_{k_0,k_1} H_{k_1,k_2} \cdots H_{k_{l-1},i_k} b_{k_l}. \tag{1.3}$$

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

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

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 $\mathbf{b}$ 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 $\tilde{pr} : S \to [0,1]$ of the random walk such that

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

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

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

F where $pr(\cdot, i) : S \to [0,1]$ $\forall i \in S$. 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 firs we introduce quantities W_j such that

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

By defining

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

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

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

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

$$E[W_i b_{k_i}] = (\mathbf{h}, H^i \mathbf{b}), \quad i = 0, 1, 2, \cdots$$

and

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

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

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

By doing so, we have

$$\theta_i = E\left[\sum_{l=0}^{\infty} W_l b_{k_l} \right] = x_i = \sum_{l=0}^{\infty} \sum_{k_1=1}^{n} \sum_{k_2=1}^{n} \cdots \sum_{k_l=1}^{n} P_{k_0,k_1} P_{k_1,k_2} \cdots P_{k_{l-1},k_l} w_{k_0,k_1} w_{k_1,k_2} \cdots w_{k_{l-1},k_l} b_{k_l}. \tag{1.4}$$

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 (1.4) 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

$$\tilde{\theta}_i = E\Big[\sum_{l=0}^{m} W_l b_{k_l}\Big] = x_i = \sum_{l=0}^{m}\sum_{k_1=1}^{n}\sum_{k_2=1}^{n}\cdots\sum_{k_l=1}^{n} P_{k_0,k_1}P_{k_1,k_2}\cdots P_{k_{l-1},k_l} w_{k_0,k_1}w_{k_1,k_2}\cdots w_{k_{l-1},k_l} b_{k_l}.$$

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 (1.1)

$$A^T\mathbf{y} = \mathbf{d}, \tag{1.5}$$

where $\mathbf{y}$ and $\mathbf{d}$ are the adjoint solution and source term. Instead A^T is the transpose matrix. By rewriting (1.5) 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

$$y_i = \sum_{l=0}^{\infty}\sum_{k_1=1}^{n}\sum_{k_2=1}^{n}\cdots\sum_{k_l=1}^{\infty} H_{k_l,k_{l-1}}\cdots H_{k_2,k_1}H_{k_1,k_0}d_{k_l}. \tag{1.6}$$

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

$$\mathbf{d} = \delta_{\mathbf{j}}$$

we have

$$(\mathbf{y}, \mathbf{b}) = (\mathbf{x}, \delta_{\mathbf{j}}) = x_j.$$

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. 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

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

The transition matrix is defined as

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

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

$$\theta_j = E\Big[\sum_{l=0}^{\infty} W_l \delta_{k_l,j}\Big] = \sum_{l=0}^{\infty}\sum_{k_1}^{n}\sum_{k_2}^{n}\cdots\sum_{k_l}^{n} b_{k_0} P_{k_0,k_1} P_{k_1,k_2}\cdots P_{k_{l-1},K_l} w_{k_0,k_1}\cdots w_{k_{l-1},k_l}\delta_{k_l,j}. \quad (1.7)$$

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

$$\tilde{\theta}_j = E\Big[\sum_{l=0}^{m} W_l \delta_{k_l,j}\Big] = \sum_{l=0}^{m}\sum_{k_1}^{n}\sum_{k_2}^{n}\cdots\sum_{k_l}^{n} b_{k_0} P_{k_0,k_1} P_{k_1,k_2}\cdots P_{k_{l-1},K_l} w_{k_0,k_1}\cdots w_{k_{l-1},k_l}\delta_{k_l,j}.$$

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

$$x_i - E\Big[\sum_{l=0}^{\infty} W_l b_{k_l}\Big] = \mathcal{O}\Big(\frac{1}{\sqrt{N}}\Big).$$

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

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

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 (1.4) and (1.7), 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.

1.4 Sequential Monte Carlo and Monte Carlo Synthetic Acceleration

However, 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. Thus it is inconvenient to use these approaches.

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 done by Halton who introduced a scheme such as the following, named *Sequential Monte Carlo*([Hal62]).

Consider e linear system such as 1.1. It can be reviewed in a fixed point scheme

$$\mathbf{x} = H\mathbf{x} + \mathbf{f}.$$

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 1: 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.

Recently Evans et al. introduced a new method which basically resorts to a Richardson scheme: 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 2: Monte Carlo Synthetic Acceleration

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

Chapter 2

Numerical results

In this section we take into account two different linear systems to analyze the properties of the main schemes we have introduced before. The first linear system is used to test the effectiveness of both Forward and Adjoint Monte Carlo methods without the help of any numerical scheme.

Then, the attention is turned on a more practical problem, a partial differential one. For this, Sequential Monte Carlo and Monte Carlo Synthetic Acceleration are used to find the solution. Both the Forward and the Adjoint Monte Carlo are employed in order to compute the updating solution at each Richardson iteration.

2.1 Test case 1

For this test case a linear system of the form $A\mathbf{x} = \mathbf{b}$ is considered, with $A \in \mathbb{R}^{50\times 50}$, $\mathbf{b} \in \mathbb{R}^{50}$. In particular, A is a tridiagonal matrix such that

$$A = \begin{bmatrix} 4 & -1 & 0 & \cdots & \cdots & 0 \\ -1 & 4 & -1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & \cdots\cdots & -1 & +4. \end{bmatrix}$$

and $\mathbf{b}$ is chosen such that its components are increasing from 1 to 50 as the index progresses the same way

$$\mathbf{b} = [1, 2, 3, \cdots, 50]^T.$$

The goal is to solve the linear system with both Forward and Adjoint Monte Carlo methods. In order to do this, we can transform the system into a fixed point scheme

$$\mathbf{x} = H\mathbf{x} + \mathbf{c}.$$

In this case H and $\mathbf{c}$ are defined such that

$$H = I - P^{-1}A, \quad \mathbf{c} = P^{-1}\mathbf{b}$$

and P is a diagonal preconditioner so that $P_{i,i} = A_{i,i}\ \forall i = 1, \cdots, 50$. The spectral radius of the iteration matrix H is $\rho(H) = 0.499051664368522$, therefore $\rho(H)^{10} = 9.581976099657400 \cdot 10^{-4}$. This justifies our choice to cut off each random walk after 10 steps both in the Forward and in the Adjoint methods.

2.1.1 Forward method results

Different numbers of random walks have been considered in order to compute the solution $\mathbf{x}$, respectively $N = 10$, $N = 100$, $N = 10000$. This to compare the trend of the error for each component with the theoretical expected behaviour imposed by the CLT. The number of steps for each random walk are 20. The global relative error associated with the MC solution $\hat{\mathbf{x}}$ with respect to the one computed by the "back-slash" command in `Matlab` is characterized by a trend shown in Figure 2.1. The relative error is conceived as

$$err_{rel} = \frac{\|\hat{\mathbf{x}} - \mathbf{x}\|_2}{\|\mathbf{x}\|_2}$$

It can be noticed that the experimental curve has the same descent as the theoretical one, accordingly to the CLT. As concerns instead the relative error associated with each component x_i, results are shown in Figure 2.2. In this case the relative error for each component is computed as

$$err_{rel,i} = \frac{|\hat{x}_i - x_i|}{|x_i|}.$$

Most of the components are characterized by a relative error with the same behaviour and it is coherent with the CLT as well. However, for some components, the relative error seems to slightly increase with an increasing number of random walks. At a first glance, it seems to be in contrast with what should happen. Even if this behaviour may appear as something weird it is not. In order to explain it let us take a practical example. Consider the standard normal distribution $\mathcal{N}(0,1)$ and let us take a certain number N of sampling points from it. Calling x_i the generic sampling point, pick the *sample mean* defined as:

$$\overline{x}_N = \frac{\sum_{i=1}^{N} x_i}{N}.$$

As $N \to \infty$, the *Strong Law of Large Numbers* (SLLN) guarantees that $\overline{x}_N$ converges pointwise to the theoretical mean. In this case the theoretical mean of the normal distribution is $\mu = 0$. Moreover the *Central Limit Theorem* (CLT) guarantees that the distribution of $\overline{x}_n$ converges in law to $\mathcal{N}(0, \frac{1}{n})$. By looking at the trend of the sample mean as $N \to \infty$, we have a behaviour such as the one shown in Figure 2.3. Since the theoretical mean is null, the trend of the sample mean coincides with the curve of the error between the sample means and the theoretical one. As it can be noticed, the error does not decrease monotonically. Actually in both the statements of SLLN and CLT nothing about the monotonicity of the error trend is announced. It means that for a number of sample points $N_1 > N_2$ we may have

$$|\overline{x}_{N_1} - \mu| > |\overline{x}_{N_2} - \mu|.$$

This is related to the fact that SLLN and CLT are just asymptotic results. In Figure 2.1 the error trend coincides with the CLT result because we are looking at the error of the entire vector. Therefore the dominating behaviour which is in the background (the asymptotycal convergence) determines the slope of the curve. Somehow, we may say that the noise associated with the oscillations of the error trend for each component is filtered. However, when we focus on a single component of the solution vector, this cannot happen anymore. We still have the guarantee that the error will asymptoticly decrease as predicted by the CLT. Nevertheless it doesn't mean that the error will decrease monotonically as the number of random walks increases.

2.1.2 Adjoint method

Looking at the relative error of the solution in Figure 2.4, we can notice that the last part of the graph, concerning the higher number of random walks, respects the theoretical expectations.

2.2 Test case 2

In this case we consider the Laplace problem: $\Omega = (0,1) \times (0,1)$

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

with $f = 1$ in Ω.

By introducing a finite difference discretization with a 5 point stencil and by setting the discretization step $h = 0.\overline{1}$, we have a 10×10 grid which reduces to a 8×8 for the homogeneous Dirichlet boundary conditions. The system we have is

$$A\mathbf{u} = \mathbf{f} \tag{2.1}$$

where $A \in \mathbf{R}^{64\times64}$, $\mathbf{u} \in \mathbb{R}^{64}$, $\mathbf{f} \in \mathbb{R}^{64}$.

The pattern of matrix A is shown in Figure 2.5 The linear system (2.1) has been solved with

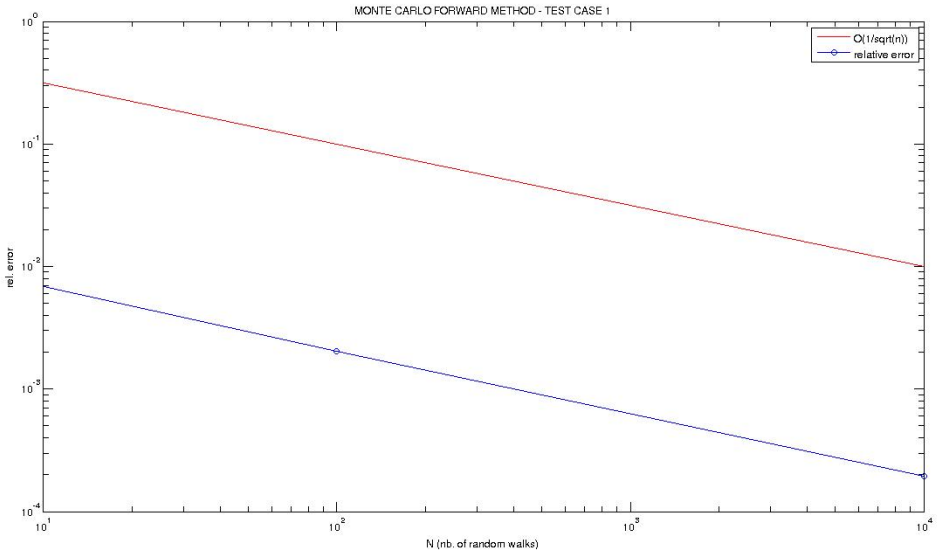


Figure 2.1: MC Forward: Relative error of the entire vector in terms of the number of random walks.

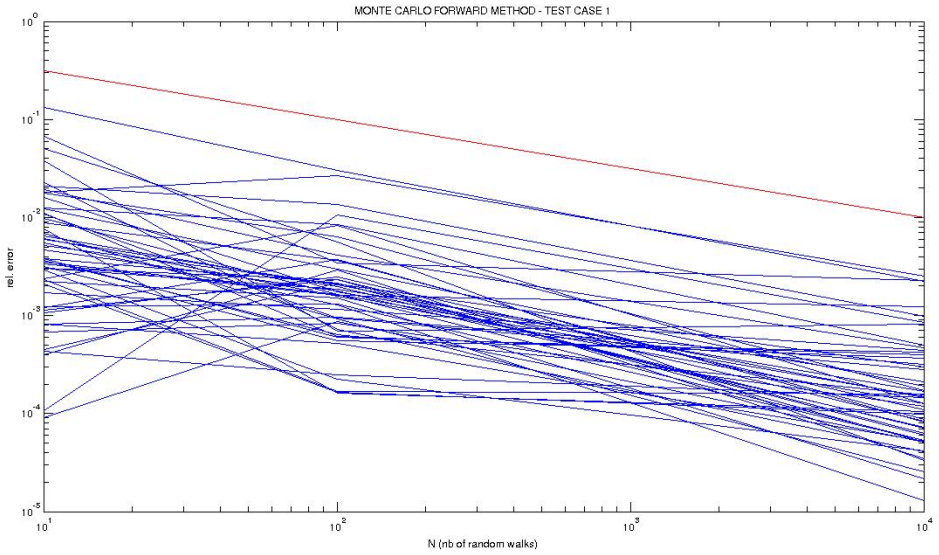


Figure 2.2: MC Forward: Relative error for each component in terms of the number of random walks.

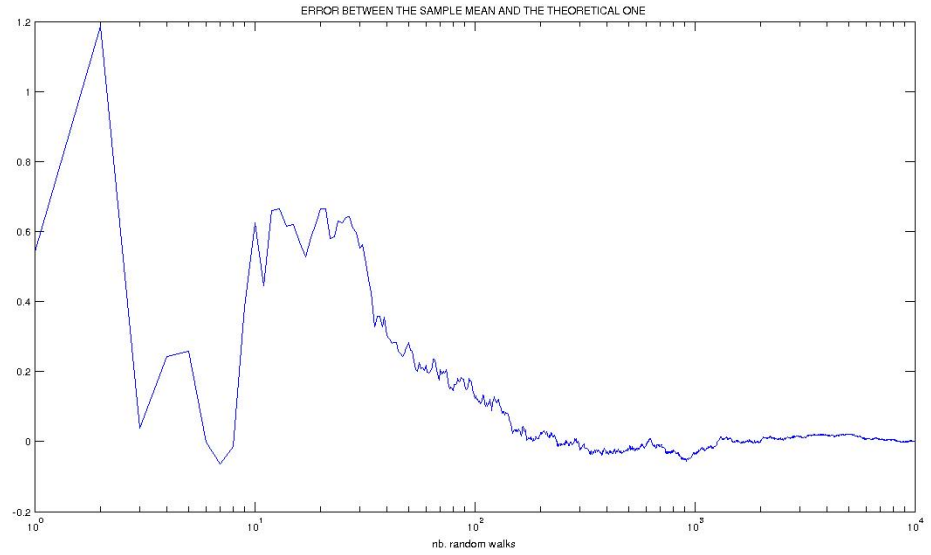


Figure 2.3: Curve of the error between the sample mean and the theoretical mean for points sampled from a standard normal distribution.

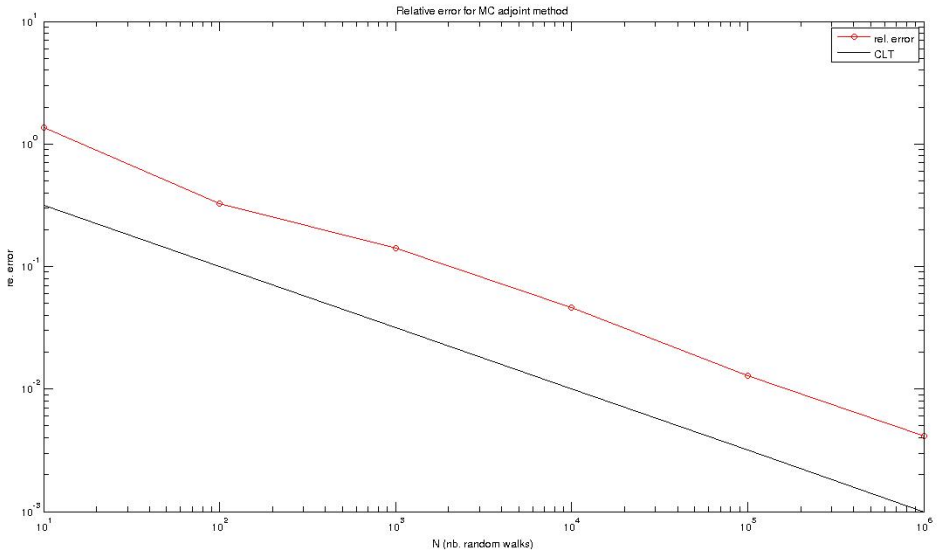


Figure 2.4: MC Adjoint: Relative error of the entire vector in terms of the number of random walks.

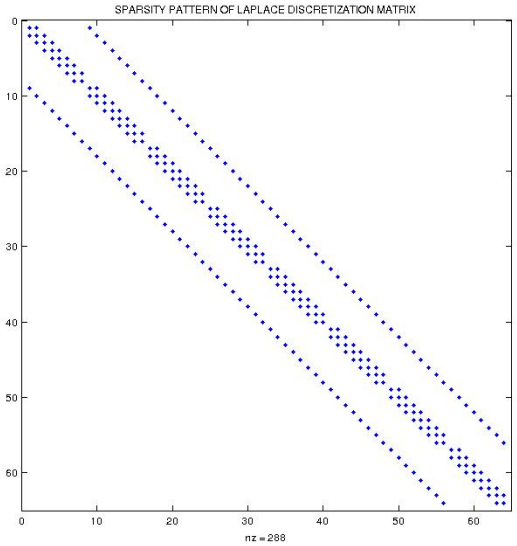


Figure 2.5: Sparsity pattern of the matrix associated with the discretization of laplacian operator.

Sequential Monte Carlo and Monte Carlo Synthetic Acceleration. Both Forward and Adjoint method are used for the computation of δx at each Richardson iteration. Moreover different trials have been done in order to modify the sparsity pattern of the matrix, testing the behaviour of the two schemes as the structure of the matrix varies. The experiments done are substantially three:

- diagonal preconditioning
- red-black reordering with Gauss-Seidel splitting
- red-black reordering with block triangular preconditioning

2.2.1 Diagonal preconditioning

By defining the iteration matrix H as for test case 1, we have $H = I - P^{-1}A$. The spectral radius of the iteration matrix is $\rho(H) = 0.94$. Therefore the Neumann-Ulam expansion is guaranteed to converge and MC can be applied inside of the Richardson scheme. For the sake of feasibility, the number of steps for each random walk is fixed again to 20. The maximal number of Richardson iteration allowed is set to 300. The tolerance is set to $\varepsilon = 10^{-3}$. The numerical setting and the results by using the Forward and the Adjoint Sequential methods are gathered in Table 2.1.

Instead the setting and results by using the Forward and the Adjoint MCSA methods are gathered in Table 2.2. It can be seen that the Adjoint method succeeds in reaching an accurate solution in a shorter amount of time. Moreover a slight improvement is noticed passing from the Sequential method to the MCSA in terms of Richardson iteration. Graphs for the correct solution and the two computed by MCSA schemes are shown in Figure 2.6 and 2.7.

	SEQ - Forward method	SEQ - Adjoint method
Nb. random walks	10^2	10^3
Numerical. it.	10	10
Relative residual	$8.21 \cdot 10^{-4}$	$7.17 \cdot 10^{-4}$
Time (s)	18.7	3.85

Table 2.1: Test case 2; Sequential Monte Carlo. Diagonal preconditioning. n=10

As follows we present results associated with the solution of the same problem by increasing the number of nodes in the grid.

By introducing $n = 22$ nodes along each direction we get $A \in \mathbb{R}^{400\times400}$, $\mathbf{f} \in \mathbb{R}^{400}$. The spectral radius of the iteration matrix now is 0.989.

	MCSA - Forward method	MCSA - Adjoint method
Nb. random walks	10^2	10^3
Numerical. it.	9	9
Relative residual	$8.95 \cdot 10^{-4}$	$8.75 \cdot 10^{-4}$
Time (s)	16.8	3.48

Table 2.2: Test case 2; MCSA. Diagonal preconditioning. n=10.

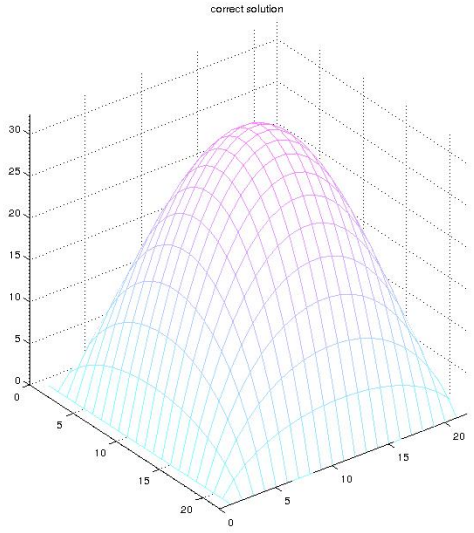


Figure 2.6: Correct solution to the Laplace problem.

As concerns the Adjoint Sequential method, the number of random walks has been incresed to 10^4. In fact this seems to be the smallest number necessary to make the algorithm converge. For the Forward method a similar modification had been apported, increasing the number of random walks employed for each entry to 20. The two approaches seem to be similar in terms of performaces. Results are represented in Table 2.3.

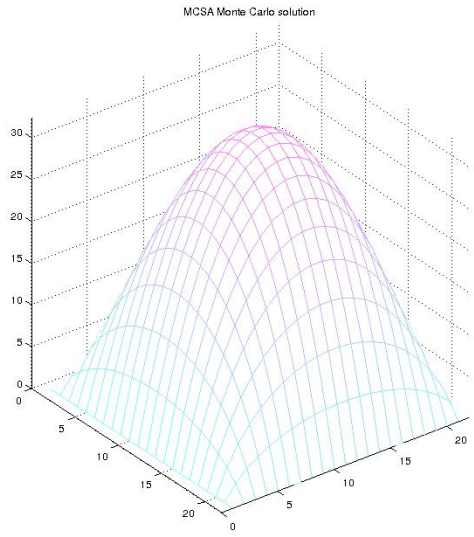
	SEQ - Forward method	SEQ - Adjoint method
Nb. random walks	20	10^4
Numerical. it.	41	30
Relative residual	$7.62 \cdot 10^{-4}$	$8.4 \cdot 10^{-4}$
Time (s)	255	407

Table 2.3: Test case 2; Sequential Monte Carlo. Diagonal preconditioning. n=22.

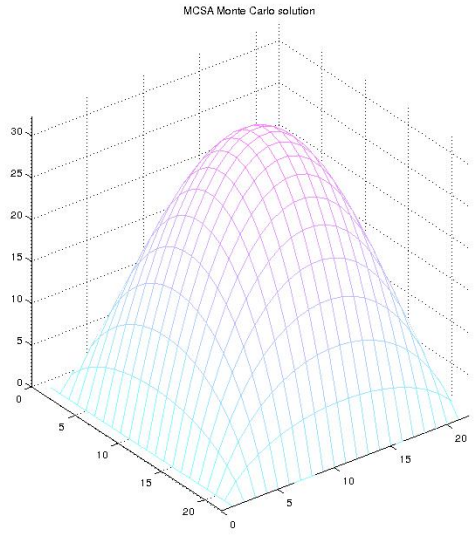
The use fo the Monte Carlo Synthetic Acceleration improves the results in terms of computational cost for both the methods, as shown in Table 2.4.

	MCSA - Forward method	MCSA - Adjoint method
Nb. random walks	10	$400 \cdot 10$
Numerical. it.	34	30
Relative residual	$8.14 \cdot 10^{-4}$	$7.89 \cdot 10^{-4}$
Time (s)	108	169

Table 2.4: Test case 2; MCSA Diagonal preconditioning. n=22.



(a) MCSA forward solution



(b) MCSA adjoint solution.

Figure 2.7: Jacobi preconditioning. MCSA with Forward scheme (top), MCSA with Adjoint scheme (bottom). $h = 0.05$

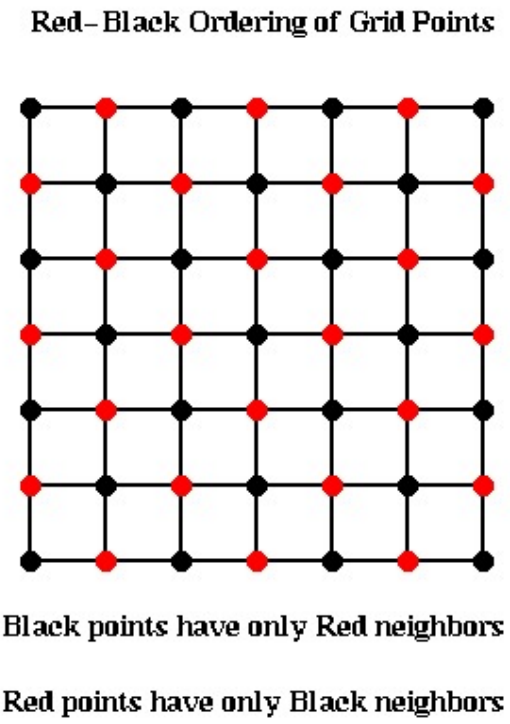


Figure 2.8: Red-black labeling for the nodes of a mesh grid.

2.2.2 Red-black reordering with Gauss-Seidel splitting

The red-black reordering consists in labeling the nodes of the mesh so that red nodes and black ones alternate each other. Red nodes must have only black neighbors and conversely. An illustrative picture is represented in Figure 2.8.

By re-enumerating the nodes of the grid so that all the red nodes are enumerated first and then all the black ones, we can consequently reorder the rows of A matrix. It implies the application of a permutation matrix $D \in \mathbb{R}^{n\times n}$ so that $D_{i,j} = 1$ if the i-th row of matrix A is sent into a j-th row of its permutation. $D_{i,j} = 0$ otherwise. The linear system we are interested in thus becomes

$$D^T AD\tilde{\mathbf{u}} = D^T\mathbf{f}.$$

The permuted matrix is $\tilde{A} = D^T AD$. The reordering of the rows changes the sparsity pattern of the matrix associated with the laplacian operator (see Figure 2.9).

The structure of matrix $\tilde{A}$ is such that

$$\tilde{A} = \begin{bmatrix} D_1 & B \\ C & D_2 \end{bmatrix}$$

By setting

$$M = \begin{bmatrix} D_1 & 0 \\ C & D_2 \end{bmatrix}$$

and

$$N = \begin{bmatrix} 0 & -B \\ 0 & 0 \end{bmatrix}$$

we have

$$\tilde{A} = M - N$$

By introducing again the fixed point representation, iteration matrix assumes this forms

$$H = M^{-1}N = \begin{bmatrix} D_1^{-1} & 0 \\ -D_2^{-1}BD_1^{-1} & D_2^{-1} \end{bmatrix} \begin{bmatrix} 0 & -B^T \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -D_1^{-1}B \\ 0 & D_2^{-1}CD_1^{-1}B \end{bmatrix}.$$

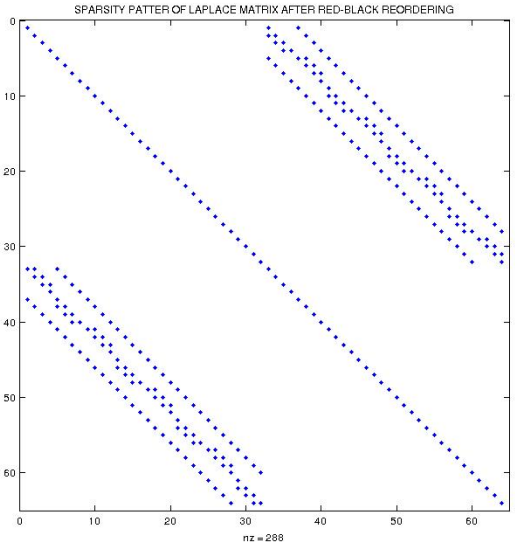


Figure 2.9: Red-black reordering of matrix laplacian matrix $\tilde{A}$.

Differently from what happened with the Jacobian preconditioning, where all the rows and all the columns have at least a term which is not null, here half of the columns contain just null elements. All the nonzero terms are clustered in the rightmost part of the matrix. Moreover we can notice that the block $D_2^{-1}CD_1^{-1}B$ is characterized by the same sparsity pattern as CB. The sparsity patterns of matrices M, N, H are represented in Figure 2.12.

The spectral radius of the iteration matrix now is $\rho(H) = 0.883$.

By resorting to Sequential and MCSA methods also in this case, we have results shown in Tables 2.5 and 2.6. The maximal number of Richardson iterations is still equal to 300, as much as the number of steps for each random walk and the tolerance.

Sequential Monte Carlo and MCSA with Adjoint take more time to compute the solution with this particular splitting. This can be justified by the fact that half of the columns of H matrix are full of null elements. Therefore the transition probability matrix, for the Adjoint method, is characterized by half of the rows that are completely null. This means that all the random walks that jump into an index from 1 to 32 at the second step are destined to stop immediately. This fact, along with the observation that in the Adjoint method the random walks is useful to estimate the entire solution vector, drives us to the conclusion that this result is reasonable. Graphs for solution with both Forward and Adjoint methods are represented in Figure 2.13.

However it is detected a significant improvement by using MCSA rather than the Sequential Monte Carlo also with the Adjoint Method.

	SEQ - Forward method	SEQ - Adjoint method
Nb. random walks	10^2	10^3
Numerical. it.	5	71
Relative residual	$9.65 \cdot 10^{-4}$	$9.97 \cdot 10^{-4}$
Time (s)	22.8	579.3

Table 2.5: Test case 2; Sequential Monte Carlo. Red-black reordering and Gauss-Seidel preconditioning. n=10.

By resorting to the solution of the Laplace problem for $n = 22$ we have results shown in Tables 2.7 and . In this case neither the Sequential Monte Carlo nor the MCSA reach convergence along with the Adjoint method. Accordingly to what already said previously about the definition of the transition probability, it further gives support to our explanation. As concerns the Forward method, instead, the scheme still succeeds in reaching convergence within the maximal number of numerical iterations.

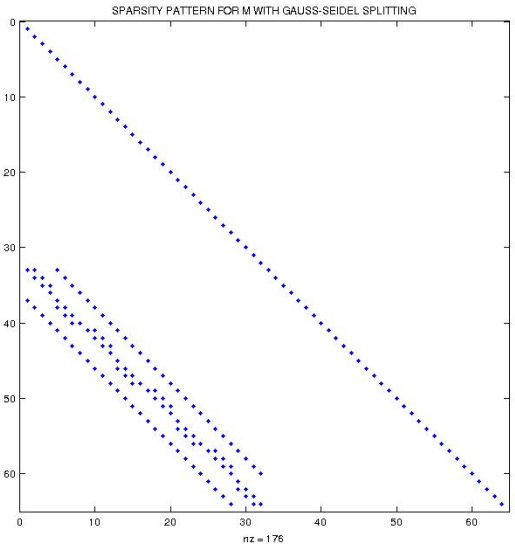


Figure 2.10: Sparsity pattern of matrix M with red-black reordering and Gauss-Seidel preconditioning.

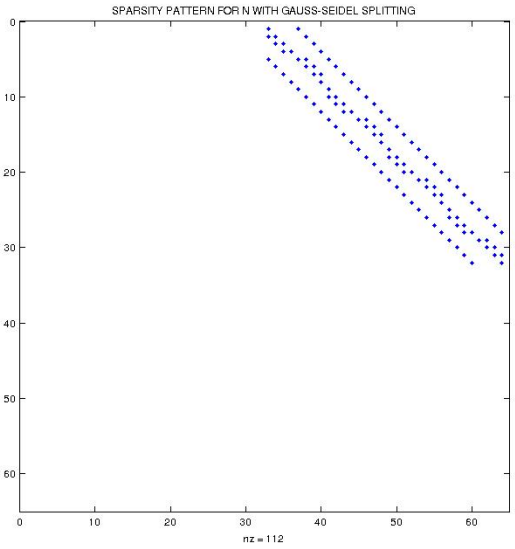


Figure 2.11: Sparsity pattern of matrix N with red-black reordering and Gauss-Seidel preconditioning.

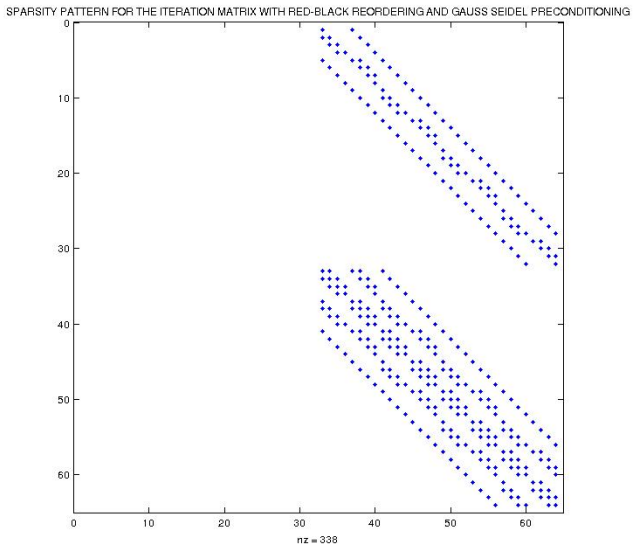


Figure 2.12: Sparsity pattern of iteration matrix with red-black reordering and Gauss-Seidel preconditioning.

	MCSA - Forward method	MCSA - Adjoint method
Nb. random walks	10^2	10^5
Numerical. it.	4	8
Relative residual	$1.22 \cdot 10^{-4}$	$3.54 \cdot 10^{-4}$
Time (s)	18.7	64.5

Table 2.6: Test case 2; MCSA. Red-black reordering and Gauss-Seidel preconditioning. n=10.

	SEQ - Forward method	SEQ - Adjoint method
Nb. random walks	10^2	10^5
Numerical. it.	16	-
Relative residual	$9.5 \cdot 10^{-4}$	-
Time (s)	624.6	-

Table 2.7: Test case 2; Sequential Monte Carlo. Red-black reordering and Gauss-Seidel preconditioning. n=22

	MCSA - Forward method	MCSA - Adjoint method
Nb. random walks	10^2	10^5
Numerical. it.	15	-
Relative residual	$9.24 \cdot 10^{-4}$	-
Time (s)	586.25	-

Table 2.8: Test case 2; MCSA. Red-black reordering and Gauss-Seidel preconditioning. n=22.

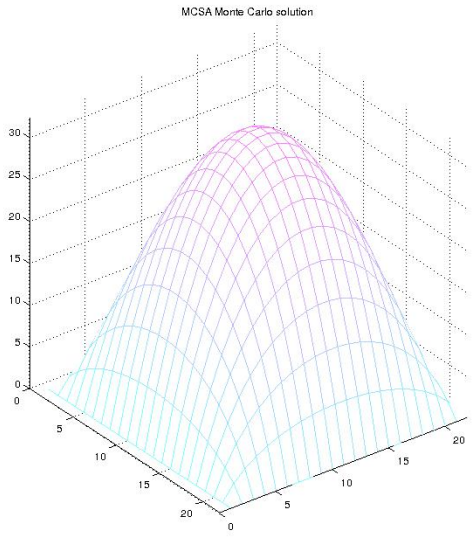
2.2.3 Red-black reordering and block triangular preconditioning

We consider a block triangular splitting such as

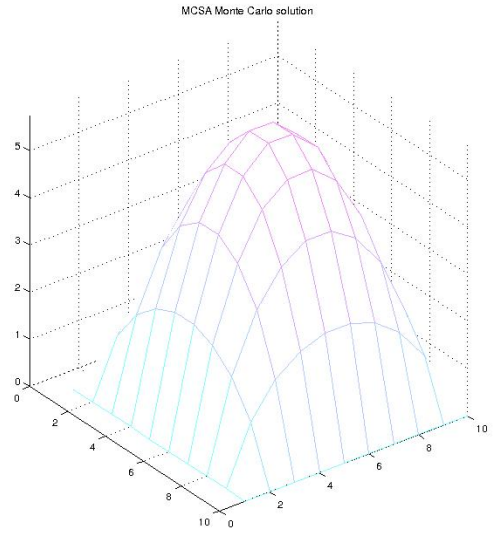
$$A = \begin{bmatrix} D_1 & B \\ C & D_2 \end{bmatrix} = \begin{bmatrix} D_1 & 0 \\ C & S \end{bmatrix} \begin{bmatrix} I & D_1^{-1}B \\ 0 & I \end{bmatrix}$$

where $S \in \mathbb{R}^{32\times 32}$ is the Schur complement defined as

$$S = D_2 - CD_1^{-1}B.$$



(a) MCSA forward solution



(b) MCSA adjoint solution.

Figure 2.13: Gauss-Seidel splitting. MCSA with Forward scheme (top), MCSA with Adjoint scheme. (bottom). top $h = 0.05$. bottom $h = 0.\overline{1}$

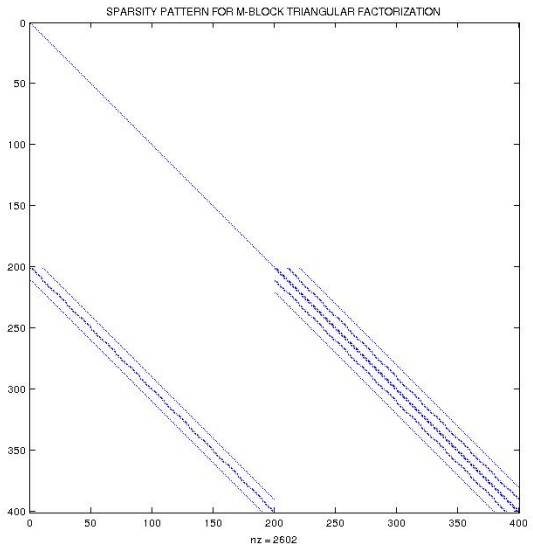


Figure 2.14: Sparsity pattern for M with triangular block factorization.

If we use

$$M = \begin{bmatrix} D_1 & 0 \\ C & S \end{bmatrix} \quad \text{s.t.} \quad A = MN$$

as a preconditioner for our linear system, then the iteration matrix is

$$H = I - M^{-1}A = \begin{bmatrix} 0 & D_1^{-1}B \\ 0 & 0 \end{bmatrix}.$$

Sparsity patterns for M, N, H are represented in Figure 2.14, 2.15, 2.16.

It is very important to stress out that in this case H is nihilpotent. In fact

$$H^2 = 0_{64\times 64}.$$

Therefore radial spectrum is $\rho(H) = 0$. This entails that the Neumann series stops at the second term

$$(I - H)^{-1} = I + H.$$

Hence we expect the MCSA methods to converge very quickly to the exact solution.

Results are shown in Table 2.9 and 2.10. Both Forward and Adjoint method apply satisfactory results inside of the Sequential and MCSA schemes. It can be seen that the number of numerical iterations employed is higher for the Adjoint method. I think this is due to the same reason that may explain inefficiency with the Gauss-Seidel preconditioning. The presence of too many columns completely null may cause the sudden death of many random walks. Thus an higher number of numerical iterations is required to cut down the residual. However the time spent is lower for the Adjoint Method than for the Forward one. Graphs with solutions are represented in Figure 2.17.

	SEQ - Forward method	SEQ - Adjoint method
Nb. random walks	10^2	10^3
Numerical. it.	2	5
Relative residual	$1.30 \cdot 10^{-16}$	$2.78 \cdot 10^{-4}$
Time (s)	0.63	0.75

Table 2.9: Test case 2; Sequential Monte Carlo. Red-black reordering and block triangular preconditioning. n=10.

We repeated the same process by setting $n = 22$ nodes along each dimension of the square domain. The performance of the Adjoint Sequential Monte Carlo is disappointing, since even in a

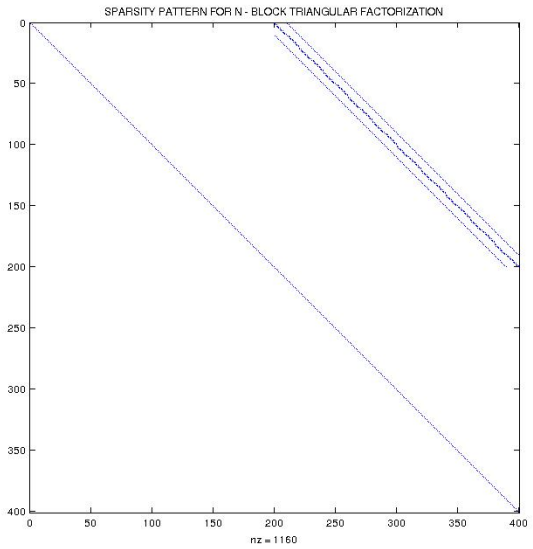


Figure 2.15: Sparsity pattern for N with triangular block factorization.

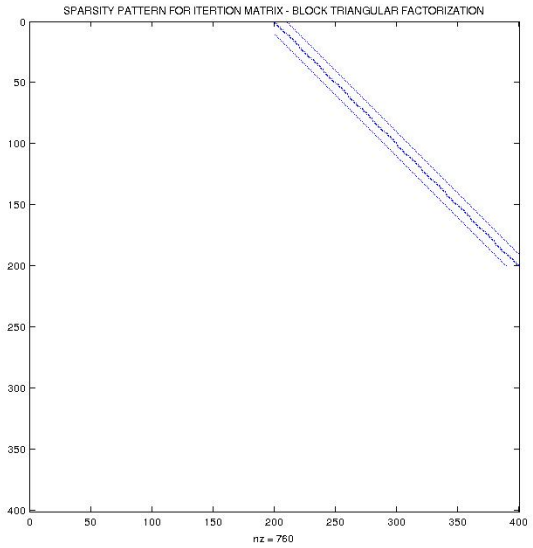


Figure 2.16: Sparsity pattern for H with triangular block factorization.

	MCSA - Forward method	MCSA - Adjoint method
Nb. random walks	10^2	10^3
Numerical. it.	1	2
Relative residual	$7.92 \cdot 10^{-16}$	$5.8 \cdot 10^{-17}$
Time (s)	0.43	0.17

Table 2.10: Test case 2; MCSA. Red-black reordering and block triangular preconditioning. n=10.

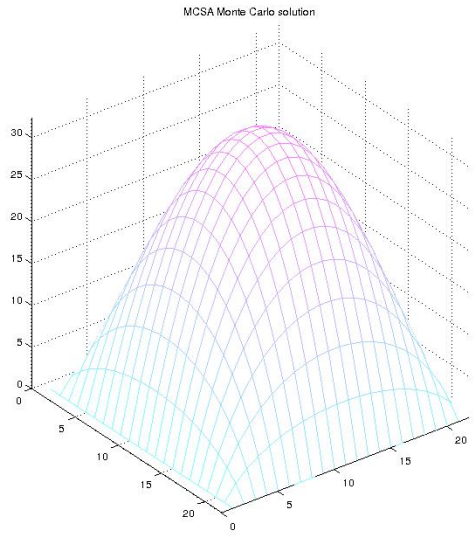
situation where the algorithm is expected to converge in a fast way thanks to the quasi-nihilpotent iteration matrix, just a relative error of 10^{-4} is given in output. The Forward Method still works fine and, as already happened for different scenarios presented in this work, a passage from the Sequential Monte Carlo to the Monte Carlo Synthetic Acceleration improves the results, especially for the Adjoint Method.

	SEQ - Forward method	SEQ - Adjoint method
Nb. random walks	10^2	10^3
Numerical. it.	2	24
Relative residual	$1.11 \cdot 10^{-16}$	$7.86 \cdot 10^{-4}$
Time (s)	4.38	2.27

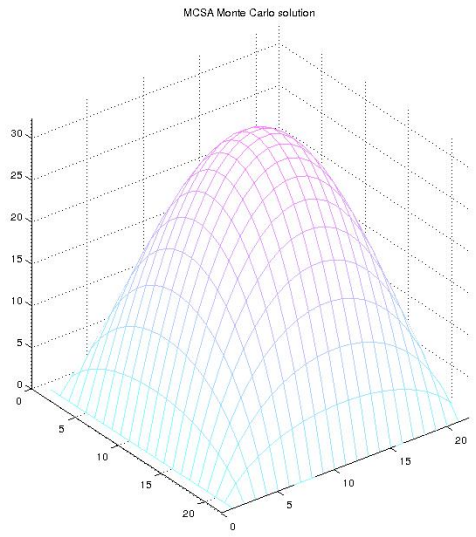
Table 2.11: Test case 2; Sequential Monte Carlo. Red-black reordering and block triangular preconditioning. n=22.

	MCSA - Forward method	MCSA - Adjoint method
Nb. random walks	10^2	10^5
Numerical. it.	1	2
Relative residual	$1.19 \cdot 10^{-15}$	$7.35 \cdot 10^{-17}$
Time (s)	1.96	0.37

Table 2.12: Test case 2; MCSA. Red-black reordering and block triangular preconditioning. n=22.



(a) MCSA forward solution



(b) MCSA adjoint solution.

Figure 2.17: Block triangular factorization. MCSA with Forward scheme (top), MCSA with Adjoint scheme (bottom). $h = 0.05$

Chapter 3

Future developments

In the near future our intentions is to test different kinds of numerical schemes, such as alternating methods ([BU07]) and implement new routines capable to explicitly compute the inverse of the system matrix A, as shown in [AAB$^+$05].

From the statistical point of view we have the intention to analyze the variance of the scheme in order to provide an interval estimation along with a pointwise one.

Bibliography

[AAB+05] V. Alessandrov, I. Atanassov, E. amd Dimov, S. Branford, A. Thandavan, and C. Weihrauch. Parallel resolvent Monte Carlo algorithms for linear algebra problems. *Lecture Notes in Computer Science*, 3516:752–759, 2005.

[BU] M. Benzi and B. Ucar. Product preconditioning for Markov chain problems.

[BU07] M. Benzi and B. Ucar. Block trianguar preconditioners for M-matrices and Markov chains. *Electronic Transaction on Numerical Analysis*, 26:209–227, 2007.

[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.

[ESW02] T.M. Evans, S.R. Slattery, and P.P.H Wilson. Mixed Monte Carlo parallel algorithms for matrix computation. *International Conference on Computational Science 2002*, 2002.

[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 Proceeding 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.