Deterministic and stochastic acceleration techniques for Richardson-type iterations

By

Massimiliano Lupo Pasini

Doctor of Philosophy

Mathematics

Deterministic and stochastic acceleration techniques for Richardson-type iterations

By

Massimiliano Lupo Pasini
B.Sc., Politecnico di Milano, 2011
M.Sc., Politecnico di Milano, 2013

Advisor: Michele Benzi, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Mathematics
2018

Abstract

Deterministic and stochastic acceleration techniques for Richardson-type iterations

By Massimiliano Lupo Pasini

Since scientific computing is involved in increasingly demanding and challenging applications from various disciplines, upcoming computing architectures are going to provide more efficient hardware resources by moving towards exascale capacities, that is $\mathcal{O}(10^{18})$ floating point operations per second (FLOPS). This computational capacity is obtained by computer manufacturers via an increase of the number of computing nodes composing the network. Therefore, the cost of global communications is expected to become more and more expensive. Because of this, the performance of state-of-the-art algorithms is expected to be highly deteriorated due to frequent global communications across the processors. On the one hand, global communications impact the computational time by constraining the parallelization. On the other hand, they make the algorithm more vulnerable to faulty phenomena (i.e. power loss, core crash, bit flips). Therefore, reducing the inter-processor communication is needed to preserve computational efficiency and reliability.

The goal of this thesis is to analyze and develop techniques to solve large scale sparse linear systems in this new computational framework. The common feature with all the approaches presented in this work is the attempt to minimize the global operations needed to solve a linear system in a multiprocessor environment. In particular, the focus of this thesis is on linear fixed point iterations, since these techniques are characterized by computational locality and simplicity. Their use to solve sparse linear systems in parallel thus opens a path towards scalability and resilience on exascale machines. However, standard fixed point schemes are renown for their deteriorated asymptotic convergence rate. To cope with this, we analyze deterministic and stochastic accelerations. The former aim to increase the efficiency by improving the convergence rate, the latter aim to enhance the robustness against faults by avoiding the propagation of locally corrupted calculations.

The deterministic acceleration we consider is the Anderson mixing. Many approaches to accelerate relaxation schemes through Anderson mixing have been studied in the literature. The approach analyzed here is called *Alternating Anderson-Richardson*. An analysis for this scheme is presented, highlighting the theoretical and computational advantages over more standard linear solvers to achieve scalability in a parallel framework. Furthermore, an augmented variant of Alternating Anderson-Richardson is proposed which guarantees convergence on positive definite linear systems.

As concerns stochastic accelerations for relaxation schemes, we analyze Monte Carlo linear solvers (MCLS) based on a stochastic reinterpretation of the fixed point algorithm. In this context, we identify classes of matrices and preconditioners that produce a convergent splitting for the stochastic fixed point iteration. Moreover, stopping criteria based on the apparent standard deviation are proposed to determine the number of statistical samplings needed to achieve a prescribed accuracy.

Deterministic and stochastic acceleration techniques for Richardson-type iterations

By

Massimiliano Lupo Pasini

B.Sc., Politecnico di Milano, 2011

M.Sc., Politecnico di Milano, 2013

Advisor: Michele Benzi, Ph.D.

A dissertation submitted to the Faculty of the

James T. Laney School of Graduate Studies of Emory University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in Mathematics

2018

Acknowledgments

I left Italy almost four years ago to come to the United States and start this wonderful and precious journey. I only had my luggage with me and many fears. Some of those fears are gone to make place to new ones. But most of them are gone to make place to unforgettable memories. Now I can say that I have more than my luggage with me.

I hope this work will inspire and help someone else in the future. Maybe another boy who leaves the past behind to start a new adventure as I did myself.

Here below I am going to mention the people I mostly feel grateful for and that contributed the most in what I have become.

First of all, I want to say ”thank you” to my Ph.D. advisor, Prof. Michele Benzi. His dedication to work, his expertise and his approach to deal with scientific topics have been a role model to me. When I was insecure about how to address an issue, he would be the kind of person that stands up from his office chair, walks up to the bookshelf, grabs a book and opens it exactly at the page where the solution to my problem is described. But I must honestly admit that the greatest gift I could receive from him was not the amount of knowledge transmitted. Indeed, I think I am still light years behind him. The greatest gift and teaching I received from my advisor is not to be afraid of what I do NOT know, so that I can approach it with enthusiasm and not being discouraged by my ignorance. Formulas, matrix properties and theorems may slip off my head. But this teaching will remain vivid in my mind.

I am also grateful to everyone from the Department of Mathematics and Computer Science at Emory University who made my graduate life as smooth and delightful as it could be. In particular, I want to thank the graduate program administrator Terry Ingram, for her patience and diligence in dealing with an absent-minded and distracted person like me.

The bitterness of my professional and personal struggles during these years has been gently sweetened by lots of friends. Some of them are consolidated friendships from the past, some of them are new ones. Among the new friendships, I want to thank all MY Latinos, who taught me how to imprint happiness in every day of my life through simple and easily applicable principles. However, a piece of my heart is preciously taken by my Italian fellows who started this trip with me many years ago back in Italy. Dear Sofia and Alessandro, your presence and your personalities completed two different sides of me: the rational one and the crazy one. Good and evil nicely coexist in equilibrium inside of me because of you. Of course, I am not necessarily associating any of you with one of these two concepts in specific (or maybe I am).

Last but not least, I want to thank my family: mamma, papà e Stefano. If I am here it is because of their sacrifice, their perseverance and their trust in my potential, even when I did not believe in myself at all. We have disagreed on many things in the past. However, I truly believe that you are my strongest support, regardless of how many oceans separate us.

Contents

Chapter 1

Introduction

Several real life applications related to disciplines such as physics and engineering require solving large scale sparse linear systems. Because of the generally high number of unknowns characterizing the linear systems of interest, it is crucial to reduce the computational time to solution. This is typically achieved in a parallel computing framework. However, as we are moving towards exascale ($\mathcal{O}(10^{18})$ FLOPS) machines, new challenges arise. On the one hand, raising the number of cores involved in a computation usually increases inter-processor communications. This could negatively impact the use of stat-of-the-art algorithms, since significant overhead may lead to inefficient performance. On the other hand, an upsurge of the number of processors increases the frequency of hard and soft system failures. Therefore, the next generation of computational science applications requires numerical solvers that are both reliable and capable of high performance on exascale platforms. To this goal, solvers need to be resilient to faulty phenomena and highly concurrent without compromising accuracy and efficiency.

As concerns the reduction of communication, standard linear solvers are not able to achieve efficiency in this new computational context. On the one hand, there are Krylov subspace methods (see Chapters 6 and 7 in [52]) that require global communications to compute inner products, which leads to inefficiency on new architectures. On the other hand, there are standard Richardson iterations (see Chapter 4 in [52]), whose convergence rate is slow and which in general may not converge. Another category of linear solvers is represented by Chebyshev iterations [43]. The advantage of these techniques is that they do not require inner products. However, they necessitate a knowledge on the spectrum of the coefficient matrix which is generally not available. Many authors have explored the opportunity to enhance the performance of Krylov methods [32,38,59]. These studies aim to reduce the inter-processor communication across iterations by relaxing the orthogonality requirement between basis vectors of the Krylov subspace. Several strategies have been employed to this goal (e.g. Block Gram-Schmidt, Tall Skinny QR, s-steps Krylov methods, delayed orthogonalization, ...). Although the expedients adopted differ for the computations being performed, they all share the goal of reconstructing the same projection subspace as standard Krylov methods in exact arithmetic. In particular, techniques addressing nonsymmetric linear systems generate the same projection subspace as the Generalized Minimal Residual method (GMRES) [53]. However, recently proposed methods to reduce communication employ accelerations based on different projection subspaces. One of these recent techniques is called *Alternating Anderson-Richardson* (AAR for short) [48,49], the main feature of which is the use of Anderson mixing [3] to accelerate standard Richardson's iteration. The literature which explains the properties of this method is not as broad and consolidated as for more standard techniques. However, a theoretical study to explain the convergence properties is provided in this thesis. Compared to other Anderson acceleration techniques [25,41,60,61], numerical experiments included in this thesis prove that AAR is more robust against stagnation. An augmented variant of this algorithm is proposed as well, namely *Augmented Alternating Anderson Richardson*, which is guaranteed to converge on linear systems with a positive definite coefficient matrix. Numerical results show AAR outperforms versions of restarted GMRES across various restarting configurations and different preconditioners. In particular, AAR successfully solves test cases where multiple versions of restarted GMRES fail to converge. Therefore, recent techniques like AAR have the potential to better handle linear systems classified as challenging for traditional linear solvers.

With regards to robustness against system failures, many modified variants of Krylov methods were proposed to accommodate resilience [1, 26, 37]. However, the presence of global operations such as inner products severely limits the level of resilience that can be obtained via Krylov methods. In particular, global operations propagate local errors across the network of interacting processors and this can severely affect the convergence of the scheme. Alternatively, one-level relaxation schemes merely employ matrix-vector multiplications to update the approximate solution. Therefore, the low amount of communication needed by Richardson's iteration opens new paths to achieve resilience. This motivated the search for resilient variant of fixed point schemes [58]. To this goal, this thesis analyzes Monte Carlo (MC) accelerations for Richardson's iteration [24, 35]. More specifically, the hybrid (deterministic-stochastic) fixed point scheme we consider is called *Monte Carlo Synthetic Acceleration* (MCSA for short). Hybrid schemes such as MCSA have been introduced to breakthrough the slow convergence of standard Monte Carlo techniques due to the Central Limit Theorem. However, the main drawback of these techniques is that the statistical estimator needs to have finite expected value and variance. These statistical conditions are algebraically reinterpreted as more restrictive requirements than the ones needed for the convergence of the deterministic Richardson's iteration. Therefore, the use of stochastic tools to accelerate Richardson's iteration reduces the set of problems upon which the fixed point scheme converges. In this thesis we identify classes of matrices and preconditioners for which convergence is guaranteed. Moreover, we propose variance-based adaptive stopping criteria to estimate the number of statistical samplings to employ to attain a prescribed accuracy.

The rest of the thesis is organized into chapters. Chapter 2 introduces the main concepts of numerical linear algebra needed to develop the discussion in the following chapters, as well as properties and algorithms of the standard linear solvers. Chapter 3 is dedicated to the Alternating Anderson-Richardson algorithm. Its main properties are presented, along with a comparison to standard linear solvers. The theoretical discussion is supported by numerical experiments at the end of this chapter, followed by conclusions on state-of-the-art and possible future developments. Chapter 4 focuses on an implementation of AAR for distributed memory parallelization using MPI as a paradigm. The goal of this chapter is to support the reason why AAR has been proposed in the literature, that is the leverage of concurrency and reduction of communication with respect to standard Krylov methods. In Chapter 5 we discuss the properties of Monte Carlo linear solvers. After a brief discussion of the contributions found in literature, we propose variance-based stopping criteria to automatically select the number of statistical samplings needed to attain a prescribed accuracy. Moreover, classes of matrices and preconditioners are identified that guarantee *a priori* the convergence of the schemes. Also in this case, the chapter is concluded with a section about numerical experiments which compare Monte Carlo techniques with standard fixed point schemes. We eventually draw conclusions about the work presented in this thesis and we propose possible future developments in Chapter 6.

Chapter 2

Standard Iterative Methods for Sparse Linear Systems

2.1 Introduction

In this chapter we present some of the standard iterative solvers that are widely employed nowadays to solve sparse linear systems. We will refer to these algorithms also in later chapters, either by presenting some new variants that improve their performance, or as a term of comparison to validate the results of our research. In particular, all the algorithms analyzed in this thesis aim to solve a square nonsingular linear system of the form

$$A\mathbf{x} = \mathbf{b}, \tag{2.1}$$

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

The chapter is structured as follows. In Section 2.2 we describe the standard Richardson scheme and we recall its convergence properties. Section 2.3 presents a broad discussion about two-level methods and all their mathematically equivalent representations in order to highlight different characteristics. Section 2.4 focuses on a specific two-level method called *Anderson-Richardson* (AR), followed by Section 2.5

where we describe the *Generalized Minimum Residual* method (GMRES). Section 2.6 compares AR and GMRES, identifying analogies and differences between the two algorithms. In conclusion Section 2.7 presents numerical experiments to support the theoretical discussion conducted in the previous sections.

2.2 The Richardson scheme

Consider a nonsingular linear system as in Equation (2.1). Applying a preconditioning matrix P^{-1} to (2.1) to the left, we have

$$P^{-1}A\mathbf{x} = P^{-1}\mathbf{b}. \tag{2.2}$$

The equation can be recast as a fixed point problem which iteratively updates the approximation to $\mathbf{x}$

$$\mathbf{x}^{k+1} = H\mathbf{x}^k + \mathbf{f}, \quad k = 0, 1, \dots, \tag{2.3}$$

where $H = I - P^{-1}A$ is the iteration matrix, $\mathbf{f} = P^{-1}\mathbf{b}$ is the preconditioned right hand side and $\mathbf{x}^0$ is an initial guess. Another equivalent representation, called correction form, is commonly adopted:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + P^{-1}\mathbf{r}^k, \quad k = 0, 1, \dots, \tag{2.4}$$

where $\mathbf{r}^k = \mathbf{b} - A\mathbf{x}^k$ is the residual at the kth iteration. A necessary and sufficient condition for the scheme in (2.4) to converge to the solution in exact arithmetic is that the spectral radius $\rho(H) < 1$. Moreover, the following relations between error and residual between consecutive iterations hold:

$$\mathbf{e}^{k+1} = H\mathbf{e}^k, \quad k = 0, 1, \dots,$$

$$\mathbf{r}^{k+1} = (I - AP^{-1})\mathbf{r}^k, \quad k = 0, 1, \ldots,$$

where the error at the kth iteration is defined as $\mathbf{e}^k = \mathbf{x} - \mathbf{x}^k$. The scheme 2.3 can be generalized by introducing a positive weighing parameter ω (namely relaxation parameter):

$$\mathbf{x}^{k+1} = (1 - \omega)\mathbf{x}^k + \omega(H\mathbf{x}^k + \mathbf{f}), \quad k = 0, 1, \ldots, \tag{2.5}$$

which in correction form is equivalently represented as

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \omega P^{-1}\mathbf{r}^k \quad k = 0, 1, \ldots \tag{2.6}$$

Equations (2.5) and (2.6) are known as preconditioned *stationary Richardson scheme* and the iteration matrix associated with this fixed point scheme is

$$H_\omega = I - \omega P^{-1}A.$$

A good choice of the matrix P^{-1} and of the relaxation parameter ω can ensure convergence in some cases (provided that all eigenvalues of A have positive real part). However, requiring that H_ω have spectral radius less than one can be too restrictive in general. Moreover, the standard one-step Richardson scheme does not efficiently damp error components related to small eigenvalues of $P^{-1}A$. Therefore, an acceleration of the scheme is recommended to increase its efficiency.

2.3 Two-Level Iterative Methods

Several mathematical contributions have been provided through the years in this respect [12, 13, 15, 39, 42, 51–53, 63]. The common idea behind these techniques is to temporarily interrupt the Richardson procedure to replace the current approximation with an improved one. The approximation is improved by solving a reduced linear system obtained by projecting the original linear system onto a subspace, motivating the use of the term *projection methods* in the mathematical literature.

From now on we refer to $A\mathbf{x} = \mathbf{b}$ as a linear system in the real field which may be already preconditioned. If p is the dimension of the *subspace of corrections* or *search subspace* $\mathcal{V}_k$ where the corrections are sought, then p constraints must be imposed to extract a corrected approximation $\overline{\mathbf{x}}^k$. A possible choice of constraints may be to enforce that the residual $\overline{\mathbf{r}}^k = \mathbf{b} - A\overline{\mathbf{x}}^k$ be orthogonal to p linearly independent vectors that identify another subspace $\mathcal{W}_k$, called *subspace of constraints* or *left subspace.* Projection methods can be categorized as *orthogonal* or *oblique* based on the way $\mathcal{V}_k$ and $\mathcal{W}_k$ relate to each other. Orthogonal projection methods create $\mathcal{W}_k$ so that it coincides with $\mathcal{V}_k$, whereas oblique projection methods use two different subspaces and $\mathcal{W}_k$ may be totally unrelated to $\mathcal{V}_k$. Projection methods can also be classified either as *multiplicative* or *additive* as in [15]. In particular, given an approximation $\mathbf{x}^k$ to the solution of (2.1), multiplicative methods are defined so as to compute an improved approximation $\overline{\mathbf{x}}^k$ as follows:

$$\text{Find} \quad \overline{\mathbf{x}}^k \in \mathcal{V}_k \quad \text{such that} \quad \mathbf{b} - A\overline{\mathbf{x}}^k \perp \mathcal{W}_k. \tag{2.7}$$

Additive methods instead search for the correction by solving the following problem:

$$\text{Find} \quad \overline{\mathbf{x}}^k \in \mathbf{x}^k + \mathcal{V}_k \quad \text{such that} \quad \mathbf{b} - A\overline{\mathbf{x}}^k \perp \mathcal{W}_k. \tag{2.8}$$

Let us denote by $\Pi_{\mathcal{V}_k} \in \mathbb{R}^{n\times n}$ an orthogonal projection operator from $\mathbb{R}^n$ onto $\mathcal{V}_k$ and $\Delta_{\mathcal{V}_k}^{\mathcal{W}_k}$ the oblique projection operator from $\mathbb{R}^n$ onto $\mathcal{V}_k$, which operates also as an orthogonal projector onto $\mathcal{W}_k$. Then the projection operators are defined by their action on a generic vector $\mathbf{v} \in \mathbb{R}^n$ as follows:

$$\Pi_{\mathcal{V}_k}\mathbf{v} \in \mathcal{V}_k, \quad \mathbf{v} - \Pi_{\mathcal{V}_k}\mathbf{v} \perp \mathcal{V}_k,$$

$$\Delta_{\mathcal{V}_k}^{\mathcal{W}_k}\mathbf{v} \in \mathcal{V}_k, \quad \mathbf{v} - \Delta_{\mathcal{V}_k}^{\mathcal{W}_k}\mathbf{v} \perp \mathcal{W}_k.$$

This allows one to recast (2.7) as

$$\text{Find} \quad \overline{\mathbf{x}}^k \in \mathcal{V}_k \quad \text{such that} \quad \Delta_{\mathcal{V}_k}^{\mathcal{W}_k}(\mathbf{b} - A\overline{\mathbf{x}}^k) = 0.$$

Since we impose $\overline{\mathbf{x}}^k \in \mathcal{V}_k$, then $\Pi_{\mathcal{V}_k}\overline{\mathbf{x}}^k = \overline{\mathbf{x}}^k$, which leads to the under-determined linear system

$$\Delta_{\mathcal{V}_k}^{\mathcal{W}_k} A\Pi_{\mathcal{V}_k}\overline{\mathbf{x}}^k = \Delta_{\mathcal{V}_k}^{\mathcal{W}_k}\mathbf{b}. \tag{2.9}$$

Likewise, the formulation in (2.8) can be equivalently written as

$$\text{Find} \quad \overline{\mathbf{x}}^k \in \mathbf{x}^k + \mathcal{V}_k \quad \text{such that} \quad \Delta_{\mathcal{V}_k}^{\mathcal{W}_k}(\mathbf{b} - A\overline{\mathbf{x}}^k) = 0.$$

For additive methods we know that the corrected approximation has the form

$$\overline{\mathbf{x}}^k = \mathbf{x}^k + \overline{\boldsymbol{\varepsilon}}^k, \quad \overline{\boldsymbol{\varepsilon}}^k \in \mathcal{V}_k.$$

Therefore, the constraint $\Delta_{\mathcal{V}_k}^{\mathcal{W}_k}(\mathbf{b} - A\overline{\mathbf{x}}^k) = 0$ still leads to the under-determined linear system (2.9). Moreover, the relation

$$\begin{aligned}\Delta_{\mathcal{V}_k}^{\mathcal{W}_k}(\mathbf{b} - A\overline{\mathbf{x}}^k) &= \Delta_{\mathcal{V}_k}^{\mathcal{W}_k}\mathbf{b} - \Delta_{\mathcal{V}_k}^{\mathcal{W}_k}(A\mathbf{x}^k) - \Delta_{\mathcal{V}_k}^{\mathcal{W}_k}(A\overline{\boldsymbol{\varepsilon}}^k)\\ &= \Delta_{\mathcal{V}_k}^{\mathcal{W}_k}\mathbf{r}^k - \Delta_{\mathcal{V}_k}^{\mathcal{W}_k}(A\overline{\boldsymbol{\varepsilon}}^k)\end{aligned}$$

turn the linear system (2.9) into

$$\Delta_{\mathcal{V}_k}^{\mathcal{W}_k} A\Pi_{\mathcal{V}_k}\overline{\boldsymbol{\varepsilon}}^k = \Delta_{\mathcal{V}_k}^{\mathcal{W}_k}\mathbf{r}^k. \tag{2.10}$$

Equations (2.9) and (2.10) are called *Petrov-Galerkin* conditions. The use of the

Petrov-Galerkin conditions allows one to reformulate (2.7) as follows:

$$\text{Find} \quad \overline{\mathbf{x}}^k \in \mathcal{V}_k \quad \text{such that} \quad \Delta_{\mathcal{V}_k}^{\mathcal{W}_k} A \Pi_{\mathcal{V}_k} \overline{\mathbf{x}}^k = \Delta_{\mathcal{V}_k}^{\mathcal{W}_k} \mathbf{b}.$$

Similarly, (2.8) can be recast as

$$\text{Find} \quad \overline{\boldsymbol{\varepsilon}}^k \in \mathcal{V}_k \quad \text{such that} \quad \Delta_{\mathcal{V}_k}^{\mathcal{W}_k} A \Pi_{\mathcal{V}_k} \overline{\boldsymbol{\varepsilon}}^k = \Delta_{\mathcal{V}_k}^{\mathcal{W}_k} \mathbf{r}^k$$

followed by $\overline{\mathbf{x}}^k = \mathbf{x}^k + \overline{\boldsymbol{\varepsilon}}^k$.

2.3.1 Matrix representation

Equations (2.9) and (2.10) represent under-determined linear systems. Therefore, the solution to these linear systems is not unique. Moreover, the requirement that $\overline{\mathbf{x}}^k \in \mathcal{V}_k$ or $\overline{\boldsymbol{\varepsilon}}^k \in \mathcal{V}_k$ is not naturally guaranteed for a generic solution to (2.9) or (2.10), respectively. To this goal, denote by $V_k \in \mathbb{R}^{n\times p}$ a matrix with column vectors that form a basis for $\mathcal{V}_k$ and $W_k \in \mathbb{R}^{n\times p}$ a matrix with column vectors that form a basis for $\mathcal{W}_k$. With regards to the multiplicative approach, the requirement that $\overline{\mathbf{x}}^k \in \mathcal{V}_k$ is restored by imposing that

$$\overline{\mathbf{x}}^k = V_k \mathbf{y}^k, \quad \mathbf{y}^k \in \mathbb{R}^p,$$

whereas the requirement that $\overline{\boldsymbol{\varepsilon}}^k \in \mathcal{V}_k$ for the additive approach is restored by enforcing that

$$\overline{\boldsymbol{\varepsilon}}^k = V_k \boldsymbol{\delta}^k, \quad \boldsymbol{\delta}^k \in \mathbb{R}^p.$$

In fact (2.7) can be recast as follows:

$$W_k^T A V_k \mathbf{y}^k = W_k^T \mathbf{b}.$$

Similarly, (2.8) can be recast as follows:

$$W_k^T A V_k \boldsymbol{\delta}^k = W_k^T \mathbf{r}^k.$$

The uniqueness of the correction can be obtained by requiring that the reduced matrix $W_k^T A V_k$ be nonsingular. In particular, the matrix $W_k^T A V_k$ is nonsingular if and only if no vector of the subspace $A\mathcal{V}_k$ is orthogonal to the subspace $\mathcal{W}_k$. Under the assumption that $W_k^T A V_k$ is nonsingular, the solution $\mathbf{x}^k$ is thus corrected in a multiplicative fashion as follows

$$\begin{aligned}\overline{\mathbf{x}}^k &= V_k \mathbf{y}^k \\ &= V_k (W_k^T A V_k)^{-1} W_k^T \mathbf{b}.\end{aligned}$$

Similarly, the additive correction is computed as

$$\begin{aligned}\overline{\mathbf{x}}^k &= \mathbf{x}^k + \overline{\boldsymbol{\varepsilon}}^k \\ &= \mathbf{x}^k + V_k \boldsymbol{\delta}^k \\ &= \mathbf{x}^k + V_k (W_k^T A V_k)^{-1} W_k^T \mathbf{r}^k.\end{aligned} \tag{2.11}$$

The specific choice of $\Pi_{\mathcal{V}_k}$ and $\Delta_{\mathcal{V}_k}^{\mathcal{W}_k}$ may heavily impact the performance of the algorithm to achieve a prescribed accuracy. This induces another categorization of the projection-iterative methods into *aggregation methods* [12,13,15,51] and *minimization methods* [15, 25, 39, 47, 60, 61]. The names are inspired by the criterion adopted to construct the projection operators $\Pi_{\mathcal{V}_k}$ and $\Delta_{\mathcal{V}_k}^{\mathcal{W}_k}$.

2.3.2 Aggregation methods

The key idea of aggregation methods consists of building two mappings S_k and P_k. In particular $S_k \in \mathbb{R}^{p\times n}$ is seen as a restriction or aggregation operator and $P_k \in \mathbb{R}^{n\times p}$

is seen as a prolongation or disaggregation operator. Assuming that

$$S_k P_k = I_p,$$

the projection operator onto the search subspace is defined as

$$\Pi_k = P_k S_k.$$

It is straightforward to check that Π_k is idempotent. Aggregation/disaggregation techniques are orthogonal projection methods. Therefore, the left subspace coincide with the search subspace, i.e.,

$$\mathcal{V}_k = \mathcal{W}_k = \mathcal{R}(\Pi_k),$$

where $\mathcal{R}(\cdot)$ is going to be used from now on to denote the range of an operator. The problem (2.9) for multiplicative aggregation/disaggregation methods thus becomes

$$\text{Find} \quad \overline{\mathbf{x}}^k \in \mathcal{R}(\Pi_k) \quad \text{such that} \quad \Pi_k A \Pi_k \overline{\mathbf{x}}^k = \Pi_k \mathbf{b}, \tag{2.12}$$

whereas the problem (2.10) now becomes

$$\text{Find} \quad \overline{\boldsymbol{\varepsilon}}^k \in \mathcal{R}(\Pi_k) \quad \text{such that} \quad \Pi_k A \Pi_k \overline{\boldsymbol{\varepsilon}}^k = \Pi_k \mathbf{r}^k, \tag{2.13}$$

followed by the additive updating step

$$\overline{\mathbf{x}}^k = \mathbf{x}^k + \overline{\boldsymbol{\varepsilon}}^k.$$

Equations (2.12) and (2.13) are called *Galerkin* conditions and they represent a specialization of (2.9) and (2.10). The solutions to (2.12) and (2.13) are not unique.

Moreover, the requirement that $\overline{\mathbf{x}}^k \in \mathcal{R}(\Pi_k)$ for the multiplicative approaches or that $\overline{\boldsymbol{\varepsilon}}^k \in \mathcal{R}(\Pi_k)$ for the additive approaches does not hold for a generic solution of (2.12) or (2.13), respectively. However, the matrix $S_k A P_k$ is nonsingular and this restores the uniqueness of the solution. In fact, with regards to multiplicative approaches, the uniqueness of the solution is obtained by solving the reduced linear system

$$S_k A P_k \mathbf{y}^k = S_k \mathbf{b},$$

followed by the prolongation step $\overline{\mathbf{x}}^k = P_k \mathbf{y}^k$. Similarly, for additive approaches the uniqueness of the solution is obtained through the reduced linear system

$$S_k A P_k \boldsymbol{\delta}^k = S_k \mathbf{r}^k,$$

followed by the prolongation step $\overline{\boldsymbol{\varepsilon}}^k = P_k \boldsymbol{\delta}^k$. Moreover, for the multiplicative approaches we obtain

$$\begin{aligned}
\Pi_k \overline{\mathbf{x}}^k &= P_k S_k \overline{\mathbf{x}}^k \\
&= P_k S_k P_k \mathbf{y}^k \\
&= P_k S_k P_k (S_k A P_k)^{-1} S_k \mathbf{b} \\
&= P_k (S_k A P_k)^{-1} S_k \mathbf{b} \\
&= P_k \mathbf{y}^k = \overline{\mathbf{x}}^k
\end{aligned} \tag{2.14}$$

which entails that $\overline{\mathbf{x}}^k \in \mathcal{R}(\Pi_k)$. Similarly, for the additive approaches we have

$$\begin{aligned}\Pi_k\overline{\boldsymbol{\varepsilon}}^k &= P_kS_k\overline{\boldsymbol{\varepsilon}}^k\\ &= P_kS_kP_k\boldsymbol{\delta}^k\\ &= P_kS_kP_k(S_kAP_k)^{-1}S_k\mathbf{r}^k\\ &= P_k(S_kAP_k)^{-1}S_k\mathbf{r}^k\\ &= P_k\mathbf{y}^k = \overline{\boldsymbol{\varepsilon}}^k, \end{aligned} \tag{2.15}$$

which implies that $\overline{\boldsymbol{\varepsilon}}^k \in \mathcal{R}(\Pi_k)$.

We report here below a local convergence analysis of aggregation methods which is carried out in [15]. The convergence analysis is conducted for multiplicative and additive methods and this will lead to an estimation of the aggregation/disaggregation acceleration rate that holds for both. We provide as follows two preliminary results used in [15] to support the following local convergence analysis. Since the authors in [15] do not provide any proofs for these results, we are going to include them here for the sake of clarity.

Proposition 1. Given $S_k \in \mathbb{R}^{p\times n}$ and $P_k \in \mathbb{R}^{n\times p}$ such that $S_kP_k = I_p$, consider the projection operator $\Pi_k = P_kS_k$. Assume that the matrix A is nonsingular and consider $H = I - A$. Then $I - \Pi_kH$ is nonsingular.

Proof. The proof is based on a contradiction argument. Let us assume that $I - \Pi_kH$ is singular. This entails that

$$[I - \Pi_kH]\tilde{\mathbf{x}} = [I - \Pi_k(I - A)]\tilde{\mathbf{x}} = 0 \tag{2.16}$$

for some $\tilde{\mathbf{x}} \neq 0$. Therefore,

$$\tilde{\mathbf{x}} = \Pi_k(I - A)\tilde{\mathbf{x}},$$

which implies that $\tilde{\mathbf{x}} \in \mathcal{R}(\Pi_k)$. The condition in (2.16) thus becomes

$$\Pi_k A \Pi_k \tilde{\mathbf{x}} = 0,$$

which leads to

$$P_k S_k A P_k S_k \tilde{\mathbf{x}} = 0.$$

Multiplying both the terms of the last equality by S_k we have

$$S_k A P_k S_k \tilde{\mathbf{x}} = 0.$$

Since $S_k A P_k$ is nonsingular, then it must be $S_k \tilde{\mathbf{x}} = 0$ which implies $\Pi_k \tilde{\mathbf{x}} = \tilde{\mathbf{x}} = 0$. However this contradicts the assumption that $\tilde{\mathbf{x}} \neq 0$. Therefore $I - \Pi_k H$ is nonsingular. $\square$

Proposition 2. Given $S_k \in \mathbb{R}^{p \times n}$ and $P_k \in \mathbb{R}^{n \times p}$ such that $S_k P_k = I_p$, consider the projection operator $\Pi_k = P_k S_k$. Assume that the matrix A is nonsingular and consider $H = I - A$. Then $I - H\Pi_k$ is nonsingular.

Proof. Let us assume that $I - H\Pi_k$ is singular. This would imply that

$$[I - H\Pi_k]\tilde{\mathbf{x}} = [I - (I - A)\Pi_k]\tilde{\mathbf{x}} = 0 \tag{2.17}$$

for some $\tilde{\mathbf{x}} \neq 0$. Therefore,

$$(I - \Pi_k)\tilde{\mathbf{x}} = -A\Pi_k \tilde{\mathbf{x}}. \tag{2.18}$$

Exploiting the fact that $(I - \Pi_k)$ is idempotent, i.e. $(I - \Pi_k)^2 = (I - \Pi_k)$, we have

$$(I - \Pi_k)\tilde{\mathbf{x}} = -(I - \Pi_k)A\Pi_k \tilde{\mathbf{x}}. \tag{2.19}$$

If we subtract Equation (2.18) from Equation (2.19) we obtain

$$\Pi_k A \Pi_k \tilde{\mathbf{x}} = 0 \quad \Rightarrow \tilde{\mathbf{x}} \in \mathcal{N}(\Pi_k),$$

where the symbol $\mathcal{N}(\cdot)$ denotes the null space of an operator. This combined with Equation (2.17) leads to

$$[I - H\Pi_k]\tilde{\mathbf{x}} = [I - (I - A)\Pi_k]\tilde{\mathbf{x}} = \tilde{\mathbf{x}} = 0,$$

which contradicts the original assumption $\tilde{\mathbf{x}} \neq 0$. Therefore $I - H\Pi_k$ is nonsingular.

$\square$

Two other important facts that we are going to use for the local convergence analysis are

$$(I - H\Pi_k)^{-1} - (I - H)^{-1} = (I - H\Pi_k)^{-1}(H\Pi_k - H)(I - H)^{-1} \tag{2.20}$$

and the fact that $(I - \Pi_k)$ is an idempotent operator. The equality (2.20) is legitimated by Proposition 2 which guarantees that the matrix $I - H\Pi_k$ is nonsingular.

Multiplicative methods

In [15] the multiplicative case is analyzed on the assumption that $\mathbf{x}^k \in \mathcal{R}(\Pi_k)$, which is equivalent to

$$\Pi_k \mathbf{x}^k = \mathbf{x}^k. \tag{2.21}$$

We describe here the convergence analysis for multiplicative aggregation/disaggregation methods by adopting the same assumption. Although this requirement may be impractical for some applications, we enforce it so as to obtain a local convergence result that holds for both multiplicative and additive approaches. Those applications for which the requirement in Equation (2.21) is not feasible can still be addressed via additive techniques. In fact, additive approaches do not require (2.21) to hold, as will be shown later in the section about additive methods.

From the equality in (2.14) for the multiplicative approach we obtain

$$\Pi_k A(\overline{\mathbf{x}}^k - \mathbf{x}^k) = \Pi_k(\mathbf{b} - A\mathbf{x}^k).$$

Therefore,

$$\begin{aligned}(I - \Pi_k H)(\overline{\mathbf{x}}^k - \mathbf{x}) &= \overline{\mathbf{x}}^k - \Pi_k H\overline{\mathbf{x}}^k - \mathbf{x} + \Pi_k H\mathbf{x} \\ &= \Pi_k\overline{\mathbf{x}}^k - \Pi_k H\overline{\mathbf{x}}^k - \mathbf{x} + \Pi_k H\mathbf{x} + \Pi_k\mathbf{x} - \Pi_k\mathbf{x} \\ &= \Pi_k(I - H)\overline{\mathbf{x}}^k - \Pi_k(I - H)\mathbf{x} + (\Pi_k - I)\mathbf{x} \\ &= \Pi_k A\overline{\mathbf{x}}^k - \Pi_k A\mathbf{x} + (\Pi_k - I)\mathbf{x} \\ &= \Pi_k\mathbf{b} - \Pi_k\mathbf{b} + (\Pi_k - I)\mathbf{x} \\ &= (\Pi_k - I)\mathbf{x}.\end{aligned}$$

Since $(I - \Pi_k H)$ is nonsingular from Proposition 1,

$$\overline{\mathbf{x}}^k - \mathbf{x} = (I - \Pi_k H)^{-1}(\Pi_k - I)\mathbf{x}. \tag{2.22}$$

By combining Equation (2.21) and Equation (2.22) we obtain

$$\overline{\mathbf{x}}^k - \mathbf{x} = (I - \Pi_k H)^{-1}(\Pi_k - I)(\mathbf{x} - \mathbf{x}^k). \tag{2.23}$$

Formula (2.23) displays the way the original approximation $\mathbf{x}^k$ is potentially improved by solving a reduced linear system on the subspace identified by the projection operator Π_k. A desirable behavior is

$$\|\overline{\mathbf{x}}^k - \mathbf{x}\| < \|\mathbf{x} - \mathbf{x}^k\|$$

with respect to some norm $\|\cdot\|$. Equation (2.23) shows that the quality of the approximation $\overline{\mathbf{x}}^k$ strongly depends on $(I - \Pi_k H)^{-1}(\Pi_k - I)$. Therefore, the projection operator Π_k plays an essential role in constructing an effective modified fixed point scheme. The benefit coming from the projected linear system may persist also on successive Richardson sweeps. This would improve the convergence rate beyond the theoretical considerations merely based on $\rho(H)$. In order to understand what properties could be required on the linear system and the projector Π_k for such a benefit, we analyze the behavior of the first Richardson sweep after the acceleration. In particular, we analyze how the aggregation/disaggregation acceleration impacts the error. We denote the first Richardson step after the aggregation-disaggregation updating as

$$\mathbf{x}^{k+1} = H\overline{\mathbf{x}}^k + \mathbf{b} = \overline{\mathbf{x}}^k + (\mathbf{b} - A\overline{\mathbf{x}}^k). \tag{2.24}$$

Recall that for the multiplicative aggregation we have $\Pi_k\overline{\mathbf{x}}^k = \overline{\mathbf{x}}^k$ and $\Pi_k(\mathbf{b} - A\overline{\mathbf{x}}^k) = 0$. Therefore from Equation (2.24) it follows that

$$\Pi_k\mathbf{x}^{k+1} = \Pi_k\overline{\mathbf{x}}^k + \Pi_k(\mathbf{b} - A\overline{\mathbf{x}}^k) = \overline{\mathbf{x}}^k,$$

which leads to

$$\mathbf{x}^{k+1} = H\overline{\mathbf{x}}^k + \mathbf{b} = H\Pi_k\mathbf{x}^{k+1} + \mathbf{b}.$$

This implies that $(I - H\Pi_k)\mathbf{x}^{k+1} = \mathbf{b}$, which together with the equality in (2.20) leads to

$$\begin{aligned}
\mathbf{x} - \mathbf{x}^{k+1} &= A^{-1}\mathbf{b} - (I - H\Pi_k)^{-1}\mathbf{b} \\
&= (I - H)^{-1}\mathbf{b} - (I - H\Pi_k)^{-1}\mathbf{b} \\
&= (I - H\Pi_k)^{-1}H(I - \Pi_k)(I - H)^{-1}\mathbf{b} \qquad (2.25) \\
&= (I - H\Pi_k)^{-1}H(I - \Pi_k)A^{-1}\mathbf{b} \\
&= (I - H\Pi_k)^{-1}H(I - \Pi_k)\mathbf{x}.
\end{aligned}$$

Because of (2.21), it holds that

$$\mathbf{x} - \mathbf{x}^{k+1} = (I - H\Pi_k)^{-1}H(I - \Pi_k)(\mathbf{x} - \mathbf{x}^k). \qquad (2.26)$$

Additive methods

As concerns the additive aggregation, we recall the equality in (2.15) and

$$\Pi_k(\mathbf{r}^k - A\overline{\boldsymbol{\varepsilon}}^k) = 0.$$

Therefore, the Galerkin property in Equation (2.13) for additive methods leads to

$$\begin{aligned}
(I - \Pi_k H)(\overline{\boldsymbol{\varepsilon}}^k - \mathbf{e}^k) &= \overline{\boldsymbol{\varepsilon}}^k - \Pi_k H\overline{\boldsymbol{\varepsilon}}^k - \mathbf{e}^k + \Pi_k H\mathbf{e}^k \\
&= \Pi_k\overline{\boldsymbol{\varepsilon}}^k - \Pi_k H\overline{\boldsymbol{\varepsilon}}^k - \mathbf{e}^k + \Pi_k H\mathbf{e}^k + \Pi_k\mathbf{e}^k - \Pi_k\mathbf{e}^k \\
&= \Pi_k(I - H)\overline{\boldsymbol{\varepsilon}}^k - \Pi_k(I - H)\mathbf{e}^k + (\Pi_k - I)\mathbf{e}^k \\
&= \Pi_k A\overline{\boldsymbol{\varepsilon}}^k - \Pi_k A\mathbf{e}^k + (\Pi_k - I)\mathbf{e}^k \\
&= \Pi_k\mathbf{r}^k - \Pi_k\mathbf{r}^k + (\Pi_k - I)\mathbf{e}^k \\
&= (\Pi_k - I)\mathbf{e}^k,
\end{aligned}$$

where $\mathbf{e}^k = \mathbf{x} - \mathbf{x}^k$ and it follows that

$$\overline{\boldsymbol{\varepsilon}}^k - \mathbf{e}^k = (I - \Pi_k H)^{-1}(\Pi_k - I)\mathbf{e}^k.$$

Making use of the following chain of equalities

$$\overline{\boldsymbol{\varepsilon}}^k - \mathbf{e}^k = \overline{\mathbf{x}}^k - \mathbf{x}^k - \mathbf{x} + \mathbf{x}^k = \overline{\mathbf{x}}^k - \mathbf{x},$$

one obtains

$$\overline{\mathbf{x}}^k - \mathbf{x} = (I - \Pi_k H)^{-1}(\Pi_k - I)(\mathbf{x} - \mathbf{x}^k).$$

Also, for additive schemes it is desirable that $\|\overline{\mathbf{x}}^k - \mathbf{x}\| < \|\mathbf{x} - \mathbf{x}^k\|$ and this strongly depends again on the matrix $(I - \Pi_k H)^{-1}(\Pi_k - I)$. Similarly to the analysis of multiplicative approaches, also for the additive approaches we aim to understand the behavior of the error associated with the first relaxation sweep after the aggregation/disaggregation step.

In the additive case, we define $\boldsymbol{\varepsilon}^{k+1} = \mathbf{x}^{k+1} - \mathbf{x}^k$. Using Equation (2.24) we obtain the following chain of equalities:

$$\begin{aligned}
\boldsymbol{\varepsilon}^{k+1} &= \mathbf{x}^{k+1} - \mathbf{x}^k \\
&= H\overline{\mathbf{x}}^k - \mathbf{x}^k + \mathbf{b} \\
&= (I - A)(\mathbf{x}^k + \overline{\boldsymbol{\varepsilon}}^k) - \mathbf{x}^k + \mathbf{b} \\
&= \mathbf{x}^k - A\mathbf{x}^k + \overline{\boldsymbol{\varepsilon}}^k - A\overline{\boldsymbol{\varepsilon}}^k - \mathbf{x}^k + \mathbf{b} \\
&= (I - A)\overline{\boldsymbol{\varepsilon}}^k + \mathbf{b} - A\mathbf{x}^k.
\end{aligned}$$

Therefore, $\boldsymbol{\varepsilon}^{k+1}$ is related to $\overline{\boldsymbol{\varepsilon}}^k$ through the following equality:

$$\boldsymbol{\varepsilon}^{k+1} = H\overline{\boldsymbol{\varepsilon}}^k + \mathbf{r}^k.$$

Combining this with the fact that $\Pi_k\overline{\boldsymbol{\varepsilon}}^k = \overline{\boldsymbol{\varepsilon}}^k$ and $\Pi_k(\mathbf{r}^k - A\overline{\boldsymbol{\varepsilon}}^k) = 0$, we have

$$\Pi_k\boldsymbol{\varepsilon}^{k+1} = \Pi_k(I - A)\overline{\boldsymbol{\varepsilon}}^k + \Pi_k\mathbf{r}^k = \Pi_k\overline{\boldsymbol{\varepsilon}}^k + \Pi_k(\mathbf{r}^k - A\overline{\boldsymbol{\varepsilon}}^k) = \overline{\boldsymbol{\varepsilon}}^k$$

and consequently

$$\boldsymbol{\varepsilon}^{k+1} = H\Pi_k \boldsymbol{\varepsilon}^{k+1} + \mathbf{r}^k,$$

which leads to $(I - H\Pi_k)\boldsymbol{\varepsilon}^{k+1} = \mathbf{r}^k$. Using the equality in (2.20) we obtain

$$\begin{aligned}
\mathbf{e}^k - \boldsymbol{\varepsilon}^{k+1} &= A^{-1}\mathbf{r}^k - (I - H\Pi_k)^{-1}\mathbf{r}^k \\
&= (I - H)^{-1}\mathbf{r}^k - (I - H\Pi_k)^{-1}\mathbf{r}^k \\
&= (I - H\Pi_k)^{-1}H(I - \Pi_k)(I - H)^{-1}\mathbf{r}^k \\
&= (I - H\Pi_k)^{-1}H(I - \Pi_k)A^{-1}\mathbf{r}^k \\
&= (I - H\Pi_k)^{-1}H(I - \Pi_k)\mathbf{e}^k.
\end{aligned}$$

Therefore, Equation (2.26) holds for the additive case as well by recalling that $\mathbf{e}^k = \mathbf{x} - \mathbf{x}^k$.

Estimation of aggregation/disaggregation accelerating factor

Since the formula in Equation (2.26) holds both for multiplicative and additive methods, it is possible to conduct a local convergence analysis of the error which applies to both types of scheme.

Let

$$\alpha_{k+1} = \frac{\|\mathbf{x} - \mathbf{x}^{k+1}\|}{\|\mathbf{x} - \overline{\mathbf{x}}^k\|}.$$

From Equation (2.23), which holds for both additive and multiplicative methods, it follows that $(\Pi_k - I)(\mathbf{x} - \mathbf{x}^k) = (I - \Pi_k H)(\overline{\mathbf{x}}^k - \mathbf{x})$. This allows one to obtain the following upper bound for the acceleration factor α_{k+1}:

$$\begin{aligned}
\alpha_{k+1} = \frac{\|\mathbf{x}-\mathbf{x}^{k+1}\|}{\|\mathbf{x}-\overline{\mathbf{x}}^k\|} &= \frac{\|(I-H\Pi_k)^{-1}H(I-\Pi_k)(\mathbf{x}-\mathbf{x}^k)\|}{\|\mathbf{x}-\overline{\mathbf{x}}^k\|} \\
&= \frac{\|(I-H\Pi_k)^{-1}H(I-\Pi_k)(I-\Pi_k)(\mathbf{x}-\mathbf{x}^k)\|}{\|\mathbf{x}-\overline{\mathbf{x}}^k\|} \\
&\leq \frac{\|(I-H\Pi_k)^{-1}H(I-\Pi_k)\|\|(I-\Pi_k)(\mathbf{x}-\mathbf{x}^k)\|}{\|\mathbf{x}-\overline{\mathbf{x}}^k\|} \\
&= \frac{\|(I-H\Pi_k)^{-1}H(I-\Pi_k)\|\|(I-\Pi_k H)(\overline{\mathbf{x}}^k-\mathbf{x})\|}{\|\mathbf{x}-\overline{\mathbf{x}}^k\|} \\
&\leq \|(I-H\Pi_k)^{-1}\|\|H(I-\Pi_k)\|\|I-\Pi_k H\|.
\end{aligned} \tag{2.27}$$

This chain of inequalities shows that a convergence rate improvement on the first Richardson sweep after the acceleration can be obtained if the quantity $\|H(I-\Pi_k)\|$ is sufficiently small. Notice that for the local convergence analysis of aggregation/disaggregation methods we have not specified which vector norm and matrix norm are adopted. The only requirement is that the matrix norm be compatible with the vector norm. The motivation behind this is due to the fact that aggregation/disaggregation accelerations are not driven by an optimality criterion in general. Therefore, the goal is not to optimize any quantity in particular. This allows one to conduct the analysis with respect to a generic norm of the error.

2.3.3 Minimization methods

We are now going to address the case when the projection operator Π_k is constructed based on some optimality criterion either for the error or for the residual. We will refer to the algorithms that follow this principle as minimization methods. The way they work is to compute a correction $\overline{\boldsymbol{\varepsilon}}^k$ that is combined with the current approximation $\mathbf{x}^k$ in an additive fashion, i.e.,

$$\overline{\mathbf{x}}^k = \mathbf{x}^k + \overline{\boldsymbol{\varepsilon}}^k.$$

Since in the rest of this work we focus on minimization methods that compute additive corrections, Equation (2.21) does not need to hold anymore. Therefore, the assumption that $\Pi_k \mathbf{x}^k = \mathbf{x}^k$ is not going to be used from now on.

The goal of minimization methods is to accelerate the Richardson iteration through techniques that minimize either the error or the residual on a subspace. The aggregation/disaggregation step is thus replaced by the solution of a least-squares problem of size $p < n$ which is applied at each iteration step. Following an error analysis presented in Chapter 4, Section 4.2 of [39], a minimization method constructs a matrix $U_k \in \mathbb{R}^{n\times p}$ at each iteration. The columns of U_k are chosen as p linearly independent vectors and the goal is to compute an improved approximation for the solution to (2.1) of the form

$$\overline{\mathbf{x}}^k = \mathbf{x}^k + U_k \mathbf{g}^k, \tag{2.28}$$

where $\mathbf{g}^k \in \mathbb{R}^p$. This is carried out in such a way that the updated error

$$\overline{\mathbf{e}}^k = \mathbf{e}^k - U_k \mathbf{g}^k$$

be reduced in some norm. Let us consider an elliptic norm, defined by

$$\|\mathbf{y}\|_G^2 = \mathbf{y}^T G \mathbf{y},$$

where $G \in \mathbb{R}^{n\times n}$ is symmetric positive definite. For a given G and a given U_k, the goal is to compute $\mathbf{g}^k$ so as to minimize the elliptic norm of the error

$$\mathbf{g}^k = \underset{\mathbf{g}\in\mathbb{R}^p}{\operatorname{argmin}} (\mathbf{e}^k - U_k \mathbf{g})^T G (\mathbf{e}^k - U_k \mathbf{g}). \tag{2.29}$$

By introducing the twice continuously differentiable convex function $\phi_k : \mathbb{R}^p \to \mathbb{R}$ defined as

$$\phi_k(\mathbf{g}) = \|\mathbf{e}^k - U_k \mathbf{g}\|_G^2, \tag{2.30}$$

$\mathbf{g}^k$ is the unique stationary point of ϕ_k and it can be computed by solving the set of p linear equations

$$\nabla_{\mathbf{g}} \phi_k(\mathbf{g}) = 0.$$

Here $\nabla_{\mathbf{g}}(\cdot)$ denotes the gradient operator with respect to the variables of the vector $\mathbf{g}$. Since

$$\nabla_{\mathbf{g}} \phi_k(\mathbf{g}) = -2U_k^T G\mathbf{e}^k + 2U_k^T G U_k \mathbf{g},$$

it follows that the vector $\mathbf{g}^k$ which minimizes the elliptic norm of the error is

$$\mathbf{g}^k = (U_k^T G U_k)^{-1} U_k^T G \mathbf{e}^k,$$

leading to an updated error such that

$$\begin{aligned}
\|\bar{\mathbf{e}}^k\|_G^2 &= (\mathbf{e}^k - U_k\mathbf{g}^k)^T G(\mathbf{e}^k - U_k\mathbf{g}^k) \\
&= (\mathbf{e}^k)^T G\mathbf{e}^k - 2(\mathbf{g}^k)^T U_k^T G\mathbf{e}^k + (\mathbf{g}^k)^T U_k^T G U_k \mathbf{g}^k \\
&= (\mathbf{e}^k)^T G\mathbf{e}^k - 2(\mathbf{e}^k)^T G U_k (U_k^T G U_k)^{-1} U_k^T G\mathbf{e}^k + (\mathbf{e}^k)^T G U_k (U_k^T G U_k)^{-1} U_k^T G\mathbf{e}^k \\
&= (\mathbf{e}^k)^T G\mathbf{e}^k - (\mathbf{e}^k)^T G U_k (U_k^T G U_k)^{-1} U_k^T G\mathbf{e}^k \\
&= \|\mathbf{e}^k\|_G^2 - (\mathbf{e}^k)^T G U_k (U_k^T G U_k)^{-1} U_k^T G\mathbf{e}^k.
\end{aligned}$$

The difference between the elliptic norms of the error before and after the updating is thus

$$\|\mathbf{e}^k\|_G^2 - \|\bar{\mathbf{e}}^k\|_G^2 = (\mathbf{e}^k)^T G U_k (U_k^T G U_k)^{-1} U_k^T G\mathbf{e}^k. \tag{2.31}$$

The matrix used to define the elliptic norm G is constant throughout all the iterations, whereas the matrix U_k is allowed to dynamically change. Without assuming any property on A besides nonsingularity, two practical choices for G are proposed in Chapter 4, Section 4.2 of [39]. The first one consists of taking $G = A^T A$, the second one consists of taking $G = I$. The former leads to solving a minimization problem for the residual ℓ^2-norm, whereas the latter leads to solving a minimization problem for the error ℓ^2-norm. Minimization methods enforce the correction $\overline{\mathbf{x}}^k$ to be as in (2.28) and computing the coefficient vector $\mathbf{g}^k$ always involves a least-squares problem. In the mathematical literature several options to choose the columns of U_k have been explored, both for the minimization of the residual and error norm. Specific practical choices in this respect will be discussed in the next sections. We remind the reader that the preliminary error analysis conducted for minimization methods already requires to consider a specific elliptic norm $\|\cdot\|_G$. This happens because minimization methods, differently from aggregation/disaggregation acceleration, are set up so as to minimize a specific norm of the error by definition. Therefore, the following error and residual analysis must be always consistent with the initial norm adopted to define the minimization technique.

Minimization of the error

Given an updated approximation as in Equation (2.28), the goal of error minimization methods is to compute $\mathbf{g}^k$ such that

$$\mathbf{g}^k = \underset{\mathbf{g}\in\mathbb{R}^p}{\text{argmin}} \|\mathbf{e}^k - U_k\mathbf{g}^k\|_2^2, \quad k = 1, 2, \dots$$

This is equivalent to computing the unique minimum point of the function ϕ_k defined in Equation (2.30) when $G = I$. In this case, the minimum is attained when

$$\mathbf{g}^k = (U_k^T U_k)^{-1} U_k^T \mathbf{e}^k.$$

However, the error $\mathbf{e}^k$ is not available. A workaround for this problem is to use a prescribed matrix $Q_k \in \mathbb{R}^{n\times p}$ such that $U_k = A^T Q_k$. Under this assumption, one has that

$$\begin{aligned}\mathbf{g}^k &= (Q_k^T A A^T Q_k)^{-1} Q_k^T A \mathbf{e}^k \\ &= (Q_k^T A A^T Q_k)^{-1} Q_k^T \mathbf{r}^k.\end{aligned}$$

Since the residual $\mathbf{r}^k$ is available, the vector $\mathbf{g}^k$ can be computed. Moreover the vector $\mathbf{g}^k$ is unique. In fact the matrix A is supposed to be nonsingular, which implies that AA^T is symmetric positive definite and this makes the matrix $(Q_k^T A A^T Q_k)^{-1}$ nonsingular as well. In this case Equation (2.31) turns into

$$\|\mathbf{e}^k\|_2^2 - \|\overline{\mathbf{e}}^k\|_2^2 = (\mathbf{r}^k)^T Q_k (Q_k^T A A^T Q_k)^{-1} Q_k^T \mathbf{r}^k$$

and Equation (2.28) becomes

$$\begin{aligned}\overline{\mathbf{x}}^k &= \mathbf{x}^k + U_k \mathbf{g}^k \\ &= \mathbf{x}^k + U_k (U_k^T U_k)^{-1} U_k^T \mathbf{e}^k \qquad (2.32) \\ &= \mathbf{x}^k + U_k (Q_k^T A U_k)^{-1} Q_k^T \mathbf{r}^k.\end{aligned}$$

Since Equation (2.32) is a specialization of Equation (2.11), we can say that error minimization methods are additive oblique projection methods where the search subspace is

$$\mathcal{V}_k = \mathcal{R}(U_k)$$

and the subspace of constraints is

$$\mathcal{W}_k = \mathcal{R}(Q_k) = A^{-T} \mathcal{R}(U_k) = A^{-T} \mathcal{V}_k.$$

Minimization of the residual

Given the residual associated with the approximation in (2.28)

$$\overline{\mathbf{r}}^k = \mathbf{b} - A \overline{\mathbf{x}}^k = \mathbf{r}^k - A U_k \mathbf{g}^k,$$

the goal of residual minimization methods is to find the optimal $\mathbf{g}^k$ that minimizes $\|\overline{\mathbf{r}}^k\|_2$, that is

$$\mathbf{g}^k = \underset{\mathbf{g}\in\mathbb{R}^p}{\operatorname{argmin}} \|\mathbf{r}^k - AU_k\mathbf{g}^k\|_2^2.$$

Since $\overline{\mathbf{r}}^k = A\overline{\mathbf{e}}^k$, then we have

$$\|\overline{\mathbf{r}}^k\|_2^2 = (\overline{\mathbf{e}}^k)^T A^T A \overline{\mathbf{e}}^k = \|\mathbf{e}^k - U_k\mathbf{g}^k\|_{A^TA}^2.$$

Therefore computing $\mathbf{g}^k$ so as to minimize $\|\overline{\mathbf{r}}^k\|_2$ results in solving the minimization problem

$$\mathbf{g}^k = \underset{\mathbf{g}\in\mathbb{R}^p}{\operatorname{argmin}} \|\mathbf{e}^k - U_k\mathbf{g}^k\|_{A^TA}^2.$$

This is equivalent to calculating the minimum point of the function ϕ_k defined in Equation (2.30) when $G = A^TA$. By defining the matrix $Z_k \in \mathbb{R}^{n\times p}$ as $Z_k = AU_k$, the minimizing vector $\mathbf{g}^k$ can be computed as

$$\begin{aligned} \mathbf{g}^k &= (U_k^T A^T A U_k)^{-1} U_k^T A^T A \mathbf{e}^k \\ &= (Z_k^T Z_k)^{-1} Z_k^T \mathbf{r}^k. \end{aligned} \tag{2.33}$$

Also in this case, the vector $\mathbf{g}^k$ is unique. In fact, the matrix A is supposed to be nonsingular, which means that A^TA is symmetric positive definite and this makes the matrix $(U_k^T A^T A U_k)^{-1}$ nonsingular as well. Equation (2.11) becomes

$$\begin{aligned} \overline{\mathbf{x}}^k &= \mathbf{x}^k + U_k\mathbf{g}^k \\ &= \mathbf{x}^k + U_k(Z_k^T Z_k)^{-1} Z_k^T \mathbf{r}^k \\ &= \mathbf{x}^k + U_k(Z_k^T A U_k)^{-1} Z_k^T \mathbf{r}^k. \end{aligned} \tag{2.34}$$

Therefore, residual minimization methods are additive oblique projection methods where the search subspace is

$$\mathcal{V}_k = \mathcal{R}(U_k)$$

and the left subspace is

$$\mathcal{W}_k = \mathcal{R}(Z_k) = A\mathcal{R}(U_k) = A\mathcal{V}_k.$$

Let us define the projection operator onto $\mathcal{R}(Z_k)$

$$\Pi_k = Z_k(Z_k^T Z_k)^{-1} Z_k^T.$$

Then Equation (2.31) becomes

$$\|\mathbf{e}^k\|_{A^TA}^2 - \|\bar{\mathbf{e}}^k\|_{A^TA}^2 = \|\mathbf{r}^k\|_2^2 - \|\bar{\mathbf{r}}^k\|_2^2 = (\mathbf{r}^k)^T Z_k(Z_k^T Z_k)^{-1} Z_k^T \mathbf{r}^k = (\mathbf{r}^k)^T \Pi_k \mathbf{r}^k. \quad (2.35)$$

It follows that

$$(\mathbf{r}^k)^T \Pi_k \mathbf{r}^k \geq 0 \quad \Rightarrow \quad \|\bar{\mathbf{e}}^k\|_{A^TA}^2 \leq \|\mathbf{e}^k\|_{A^TA}^2$$

and

$$\bar{\mathbf{r}}^k = \mathbf{b} - A\bar{\mathbf{x}}^k = \left[\mathbf{r}^k - Z_k(Z_k^T Z_k)^{-1} Z_k^T \mathbf{r}^k\right] = (I - \Pi_k)\mathbf{r}^k, \quad (2.36)$$

which implies

$$\|\bar{\mathbf{r}}^k\|_2 \leq \|\mathbf{r}^k\|_2.$$

This entails that the residual minimization correction guarantees a local non-increasing trend for the residual. Furthermore, the ability of Π_k to identify a subspace containing $\mathbf{r}^k$ affects the performance. In fact, it is preferable that $\Pi_k \mathbf{r}^k \approx \mathbf{r}^k$.

Anderson mixing

A specific choice to construct the matrix U_k for residual minimization methods has been proposed by Anderson in [3]. The technique presented by Anderson aims at computing a vector

$$\mathbf{g}^k = (g_1^k, \ldots, g_p^k)^T \in \mathbb{R}^p$$

to correct the approximation $\mathbf{x}^k$ by specializing Equation (2.28) as follows:

$$\overline{\mathbf{x}}^k = \mathbf{x}^k - \sum_{j=1}^{p} g_j^k(\mathbf{x}^{k-p+j} - \mathbf{x}^{k-p+j-1}). \tag{2.37}$$

Therefore, Equation (2.37) corrects $\mathbf{x}^k$ via a linear combination that involves the approximations computed at the previous p steps. The procedure proposed in Equation (2.37) to compute $\overline{\mathbf{x}}^k$ is called *Anderson mixing*. Define

$$X_k = [(\mathbf{x}^{k-p+1} - \mathbf{x}^{k-p}), (\mathbf{x}^{k-p+2} - \mathbf{x}^{k-p+1}), \ldots, (\mathbf{x}^k - \mathbf{x}^{k-1})] \in \mathbb{R}^{n\times p} \tag{2.38}$$

and

$$R_k = [(\mathbf{r}^{k-p+1} - \mathbf{r}^{k-p}), (\mathbf{r}^{k-p+2} - \mathbf{r}^{k-p+1}), \ldots, (\mathbf{r}^k - \mathbf{r}^{k-1})] \in \mathbb{R}^{n\times p}. \tag{2.39}$$

Under the assumption that R_k has linearly independent columns, we can recast Equation (2.37) as

$$\overline{\mathbf{x}}^k = \mathbf{x}^k - X_k\mathbf{g}^k$$

where the optimal vector $\mathbf{g}^k$ is computed as

$$\begin{aligned}\mathbf{g}^k &= \underset{\mathbf{g}\in\mathbb{R}^p}{\operatorname{argmin}} \|\mathbf{b} - A(\mathbf{x}^k - X_k\mathbf{g})\|_2^2 \\ &= (R_k^T R_k)^{-1} R_k^T \mathbf{r}^k.\end{aligned}$$

The eventuality of linearly dependent columns of R_k is discussed in Chapter 3, Remark 4.

We stress the fact that $U_k = -X_k$ and $Z_k = R_k = AU_k = -AX_k$. In particular,

Equation (2.34) becomes

$$\begin{aligned}\overline{\mathbf{x}}^k &= \mathbf{x}^k - X_k\mathbf{g}^k \\ &= \mathbf{x}^k - X_k(R_k^TR_k)^{-1}R_k^T\mathbf{r}^k \\ &= \mathbf{x}^k + X_k(R_k^TAX_k)^{-1}R_k^T\mathbf{r}^k.\end{aligned} \tag{2.40}$$

Therefore, the subspace of corrections becomes

$$\mathcal{V}_k = \mathcal{R}(X_k)$$

and the subspace of constraints is

$$\mathcal{W}_k = \mathcal{R}(R_k) = A\mathcal{R}(X_k) = A\mathcal{V}_k.$$

Remark 1. The Anderson mixing is sometimes described in the literature in a different form than Equation (2.37). In fact, an alternative definition is

$$\overline{\mathbf{x}}^k = \mathbf{x}^{k-p} - \sum_{j=1}^{p} \hat{g}_j^k(\mathbf{x}^{k-p+j} - \mathbf{x}^{k-p}), \tag{2.41}$$

where $\hat{\mathbf{g}}^k = (\hat{g}_1^k, \dots, \hat{g}_p^k)^T \in \mathbb{R}^p$. By defining the matrices $\hat{X}_k \in \mathbb{R}^{n\times p}$ and $\hat{R}_k \in \mathbb{R}^{n\times p}$ as follows

$$\hat{X}_k = [(\mathbf{x}^{k-p+1} - \mathbf{x}^{k-p}), (\mathbf{x}^{k-p+2} - \mathbf{x}^{k-p}), \dots, (\mathbf{x}^k - \mathbf{x}^{k-p})] \in \mathbb{R}^{n\times p}, \tag{2.42}$$

$$\hat{R}_k = [(\mathbf{r}^{k-p+1} - \mathbf{r}^{k-p}), (\mathbf{r}^{k-p+2} - \mathbf{r}^{k-p}), \dots, (\mathbf{r}^k - \mathbf{r}^{k-p})] \in \mathbb{R}^{n\times p}, \tag{2.43}$$

Equation (2.41) is recast as

$$\overline{\mathbf{x}}^k = \mathbf{x}^{k-p} - \hat{X}_k\hat{\mathbf{g}}^k,$$

where the optimal vector $\hat{\mathbf{g}}^k$ is computed as

$$\begin{aligned}\hat{\mathbf{g}}^k &= \underset{\mathbf{g}\in\mathbb{R}^p}{\operatorname{argmin}} \|\mathbf{b} - A(\mathbf{x}^{k-p} - \hat{X}_k\mathbf{g})\|_2^2 \\ &= (\hat{R}_k^T\hat{R}_k)^{-1}\hat{R}_k^T\mathbf{r}^{k-p}.\end{aligned}$$

Nevertheless the alternative definition of Anderson mixing in Equation (2.41) cannot be seen as a specialization of Equation (2.28).

2.4 Anderson-Richardson method

An example of residual minimization methods is known in the literature as *Anderson-Richardson* method (shortly named AR) [25,41,47,61] whose pseudo-code is described in Algorithm 1. The main idea behind this approach is to compute an Anderson mixing to accelerate the Richardson scheme at each iteration. Therefore, the algorithm alternates a Richardson sweep with a minimization problem that involves the new update computed through Richardson and the previous p approximations. Although the extrapolation method of Anderson is generally used to accelerate non-linear fixed point iterations [3], it has been proved to effectively speed-up also linear fixed point schemes on a wide range of problems [47–49, 61].

For the time being, we are going to describe the Anderson-Richardson method by assuming that the Anderson mixing is implemented according to Equation (2.37). An iteration of Anderson-Richardson that updates $\mathbf{x}^k$ into $\mathbf{x}^{k+1}$ can be decomposed into two consecutive steps as follows:

$$\begin{cases}\overline{\mathbf{x}}^k = \mathbf{x}^k - X_k(R_k^TR_k)^{-1}R_k^T\mathbf{r}^k & \text{(Anderson mixing)} \quad (2.44a)\\ \mathbf{x}^{k+1} = \overline{\mathbf{x}}^k + \beta_k(\mathbf{b} - A\overline{\mathbf{x}}^k) & \text{(Richardson sweep),} \quad (2.44b)\end{cases}$$

Algorithm 1: Anderson-Richardson (AR)

```
   Data: $\mathbf{x}^0$, $\{\beta_k\}_{k=0}^{k=\text{maxit}}$, $p$, $tol$
   Result: $\mathbf{x}^{k+1}$
 1 Compute $\mathbf{r}^0 = \mathbf{b} - A\mathbf{x}^0$;
 2 Compute $\mathbf{x}^1 = (I - \beta_0 A)\mathbf{x}^0 + \mathbf{b}$;
 3 Set $k = 1$;
 4 while $\frac{\|\mathbf{r}^{k-1}\|_2}{\|\mathbf{b}\|_2} > tol$ do
 5     Compute $\ell = \min\{k, p\}$;
 6     Compute $\mathbf{r}^k = \mathbf{b} - A\mathbf{x}^k$;
 7     Set $X_k = [(\mathbf{x}^{k-\ell+1} - \mathbf{x}^{k-\ell}), \dots, (\mathbf{x}^k - \mathbf{x}^{k-1})] \in \mathbb{R}^{n\times\ell}$;
 8     Set $R_k = [(\mathbf{r}^{k-\ell+1} - \mathbf{r}^{k-\ell}), \dots, (\mathbf{r}^k - \mathbf{r}^{k-1})] \in \mathbb{R}^{n\times\ell}$;
 9     Determine $\mathbf{g}^k = [g_1^k, \dots, g_\ell^k]^T$ such that $\mathbf{g}^k = \underset{\mathbf{g}\in\mathbb{R}^\ell}{\operatorname{argmin}} \|\mathbf{r}^k - R_k\mathbf{g}\|_2$;
10     Set $\overline{\mathbf{x}}^k = \mathbf{x}^k - X_k\mathbf{g}^k$;
11     Set $\mathbf{x}^{k+1} = \overline{\mathbf{x}}^k + \beta_k(\mathbf{b} - A\overline{\mathbf{x}}^k)$;
12     $k = k + 1$.
13 end
```

where $\beta_k \in \mathbb{R}$, $\forall k \geq 0$. An equivalent expression that relates $\mathbf{x}^{k+1}$ directly to $\mathbf{x}^k$ is

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \beta_k\mathbf{r}^k - (X_k + \beta_k R_k)(R_k^T R_k)^{-1} R_k^T \mathbf{r}^k. \tag{2.45}$$

Therefore, Anderson-Richardson can be re-interpreted as a one level non-stationary Richardson iteration

$$\mathbf{x}^{k+1} = \mathbf{x}^k + C_k\mathbf{r}^k,$$

where the scalar weight ω of Equation (2.6) is replaced with the matrix

$$C_k = \beta_k I - (X_k + \beta_k R_k)(R_k^T R_k)^{-1} R_k^T.$$

Note that the matrix C_k is dynamically updated at each iteration of Anderson-Richardson.

It is worth noticing that the monotonicity of the residual through the Anderson mixing is not enough to guarantee the convergence of Anderson-Richardson overall. In fact, the Richardson sweep of Equation (2.44b) does not usually reduce either the norm of the residual or the norm of the error.

Anderson-Richardson and the additive aggregation/disaggregation methods of Section 2.3.2 have many similarities. In fact, they are all classified as projection methods and they can be recast as two-level fixed point schemes. As already mentioned, the common feature is the presence of an underlying Richardson iteration that is periodically accelerated through an additive correction, which is computed by projecting the original linear system (2.1) onto a subspace. In order to understand better the analogy, we focus on the sequence of residual vectors generated by Anderson-Richardson. The residual associated with Equation (2.44b) is

$$\begin{aligned} \mathbf{r}^{k+1} &= (I - \beta_k A)\overline{\mathbf{r}}^k \\ &= (I - \beta_k A)(I - \Pi_k)\mathbf{r}^k. \end{aligned} \tag{2.46}$$

The effectiveness of Anderson-Richardson in reducing the residual between two consecutive iterations is thus related to the matrix $(I - \beta_k A)(I - \Pi_k)$. The more contractive it can be on $\mathbf{r}^k$, the better. Therefore, we can conclude that the matrix $(I - \beta_k A)(I - \Pi_k)$ impacts the performance of Anderson-Richardson to the same extent to which the matrix $(I - A)(I - \Pi_k)$ impacts the performance of aggregation/disaggregation methods.

We remind the reader that an alternative definition of Anderson-Richardson would be possible if the Anderson mixing were implemented as in Equation (2.41). In this case the iteration of Anderson-Richardson would become

$$\begin{cases} \overline{\mathbf{x}}^k = \mathbf{x}^{k-p} - \hat{X}_k(\hat{R}_k^T\hat{R}_k)^{-1}\hat{R}_k^T\mathbf{r}^{k-p} & \text{(Anderson mixing)} \qquad (2.47\text{a}) \\ \mathbf{x}^{k+1} = \overline{\mathbf{x}}^k + \beta_k(\mathbf{b} - A\overline{\mathbf{x}}^k) & \text{(Richardson sweep).} \qquad (2.47\text{b}) \end{cases}$$

However, in this case it is not possible to find a one step reinterpretation of the scheme that directly relates $\mathbf{x}^{k+1}$ to $\mathbf{x}^k$ as in Equation (2.45). The definition of Anderson mixing as in Equation (2.41) is usually adopted to reinterpret Anderson-Richardson as a multi-secant method [25] and to analyze the connection between Anderson-Richardson and Full GMRES. This connection is the object of the next section.

2.5 Generalized Minimum Residual Method

Given the Krylov subspace

$$\mathcal{K}_k(A, \mathbf{r}^0) = \operatorname{span}\{\mathbf{r}^0, A\mathbf{r}^0, \ldots, A^{k-1}\mathbf{r}^0\}, \tag{2.48}$$

the generalized minimum residual method (GMRES) is another example of an oblique projection method based on taking the search subspace as the Krylov subspace itself

$$\mathcal{V}_k = \mathcal{K}_k(A, \mathbf{r}^0)$$

and the left subspace as

$$\mathcal{W}_k = A\mathcal{V}_k = A\mathcal{K}_k(A, \mathbf{r}^0).$$

The GMRES method provides approximate solutions to (2.1) of the form

$$\mathbf{x}_G^k = \mathbf{x}^0 + \mathbf{z}^k,$$

where

$$\mathbf{z}^k = \underset{\mathbf{z} \in \mathcal{K}_k(A, \mathbf{r}^0)}{\operatorname{argmin}} \|\mathbf{b} - A(\mathbf{x}^{\mathbf{0}} + \mathbf{z})\|_2. \tag{2.49}$$

Let us denote by $P_{A\mathcal{K}_k}$ the orthogonal projector onto the subspace $A\mathcal{K}_k(A, \mathbf{r}^0)$. From Equation (2.49) it follows that

$$A(\mathbf{x}_G^k - \mathbf{x}^0) = A\mathbf{z}^k = P_{A\mathcal{K}_k}(\mathbf{b} - A\mathbf{x}^0) = P_{A\mathcal{K}_k}\mathbf{r}^0.$$

Therefore, $\mathbf{z}^k = A^{-1}P_{A\mathcal{K}_k}\mathbf{r}^0$ and

$$\begin{aligned} \mathbf{x}_G^k - \mathbf{x} &= \mathbf{x}^0 + A^{-1}P_{A\mathcal{K}_k}\mathbf{r}^0 - \mathbf{x} \\ &= A^{-1}[A(\mathbf{x}^0 - \mathbf{x})] + A^{-1}P_{A\mathcal{K}_k}\mathbf{r}^0 \\ &= -A^{-1}(I - P_{A\mathcal{K}_k})\mathbf{r}^0. \end{aligned} \tag{2.50}$$

Following the matrix representation of an additive method in Section 2.3.1, the method can be expressed as in formula (2.11). However, this mathematical representation is impractical to attain an efficient performance. Indeed, the size of the matrix $W_k^T A V_k$ increases along with k, making the computation of the updating vector more and more expensive. Because of this, an alternative representation is generally adopted which resorts to the Arnoldi's procedure to construct an orthogonal basis for the Krylov subspace $\mathcal{K}_k(A, \mathbf{r}^0)$. We report the steps of this orthogonalization process in Algorithm 2. Although there are multiple approaches to apply the orthogonalization step at each iteration of Arnoldi's procedure, we present only the version of the algorithm based on modified Gram-Schmidt (MGS). Other alternative implementations may resort to Householder reflections or intermediate re-orthogonalizations.

Algorithm 2: Arnoldi MGS

```
  Data: $\mathbf{v}^1$, $A$
  Result: $\{\mathbf{w}_i\}_{i=1}^k$
1 Compute $\mathbf{w}_j = A\mathbf{v}_j$;
2 for $j = 1, 2, \dots, k$ do
3     $h_{ij} = (\mathbf{w}_j, \mathbf{v}_i)$;
4     $\mathbf{w}_j = \mathbf{w}_j - h_{ij}\mathbf{v}_i$
5 end
6 Compute $h_{j+1,j} = \|\mathbf{w}_j\|_2$. If $h_{j+1,j} = 0$ Stop
7 Compute $\mathbf{v}_{j+1} = \frac{\mathbf{w}_j}{h_{j+1,j}}$.
```

If we denote by $\overline{H}_k$ the $(k+1) \times k$ Hessenberg matrix whose nonzero entries are defined by Algorithm 2 and by H_k the matrix obtained from $\overline{H}_k$ by deleting the last row, then the following relations hold:

$$AV_k = V_k H_k + \mathbf{w}^k (\mathbf{e}^k)^T \tag{2.51}$$

$$= V_{k+1} \overline{H}_k, \tag{2.52}$$

$$V_k^T A V_k = H_k. \tag{2.53}$$

The second way to derive the algorithm of GMRES is to exploit Equation (2.49) combined with Equation (2.52). Any vector $\mathbf{x}$ in the affine subspace $\mathbf{x}^0 + \mathcal{K}_k(A, \mathbf{r}^0)$ can be written as

$$\mathbf{x} = \mathbf{x}_0 + V_k \mathbf{y},$$

where $\mathbf{y} \in \mathbb{R}^k$. Therefore, using Equation 2.52 we have

$$\begin{aligned} \mathbf{b} - A\mathbf{x} &= \mathbf{b} - A(\mathbf{x}^0 + V_k \mathbf{y}) \\ &= \mathbf{r}^0 - AV_k \mathbf{y} \\ &= \beta \mathbf{v}_1 - V_{k+1} \overline{H}_{k+1} \mathbf{y} \\ &= V_{k+1}(\beta \mathbf{e}^1 - \overline{H}_k \mathbf{y}), \end{aligned}$$

where $\mathbf{v}_1$ denotes the first column of the matrix V_k and $\mathbf{e}^1$ is the first vector of the standard basis. Since the columns of the matrix V_{k+1} are orthonormal, then

$$\|\mathbf{b} - A(\mathbf{x}^0 + V_k \mathbf{y})\| = \|\beta \mathbf{e}^1 - \overline{H}_k \mathbf{y}\|_2. \tag{2.54}$$

Combining Equation (2.49) and (2.54), we have that

$$\mathbf{x}_G^k = \mathbf{x}^0 + V_k \mathbf{y}_k,$$

where

$$\mathbf{y}_k = \underset{\mathbf{y}\in\mathbb{R}^k}{\operatorname{argmin}} \|\beta\mathbf{e}^1 - \overline{H}_k\mathbf{y}\|_2. \tag{2.55}$$

The least-squares problem to solve in Equation (2.55) is usually cheap to compute, thanks to the upper Hessenberg structure of $\overline{H}_k$ and the usually small value of k to reach a prescribed accuracy. The pseudo-code for GMRES recast in this framework is shown in Algorithm 3.

Algorithm 3: GMRES

```
   Data: $\mathbf{x}^0$, $tol$
   Result: $\mathbf{x}^m$
 1 Set $k = 1$;
 2 Compute $\mathbf{r}^0 = \mathbf{b} - A\mathbf{x}^0$;
 3 Compute $\beta = \|\mathbf{r}^0\|_2$;
 4 Compute $\mathbf{v}^1 = \frac{\mathbf{r}^0}{\beta}$;
 5 while $\frac{\|\mathbf{r}^{k-1}\|_2}{\|\mathbf{b}\|_2} > tol$ do
 6     Compute $\mathbf{w}_k = A\mathbf{v}_k$;
 7     for $i = 1, \dots, k$ do
 8         $h_{ik} = \langle \mathbf{w}_k, \mathbf{v}_i \rangle$;
 9         $\mathbf{w}_k = \mathbf{w}_k - h_{ik}\mathbf{v}_i$;
10     end
11     $h_{k+1,k} = \|\mathbf{w}_k\|_2$. If $h_{k+1,k} = 0$ set $m = k$ and go to 20;
12     $\mathbf{v}_{k+1} = \frac{\mathbf{w}_k}{h_{k+1,k}}$;
13     Define the $(k+1) \times k$ Hessenberg matrix $\overline{H}_k = \{h_{ij}\}_{1\le i\le k, 1\le j\le k}$;
14     Compute the QR factorization of $\overline{H}_k$ via Givens rotations: $\overline{H}_k = Q_k\overline{R}_k$;
15     Compute $\overline{\mathbf{d}}^k \in \mathbb{R}^{k+1}$ such that $\overline{\mathbf{d}}^k = Q_k^T(\beta\mathbf{e}_1)$;
16     Compute $\|\mathbf{r}^k\|_2 = |\overline{\mathbf{d}}^k_{k+1}|$;
17     $k = k + 1$;
18 end
19 $m = k - 1$;
20 Compute $\mathbf{y}_m$, the minimizer of $\|\beta\mathbf{e}_1 - \overline{H}_m\mathbf{y}\|_2$, and $\mathbf{x}_m = \mathbf{x}_0 + V_m\mathbf{y}_m$.
```

2.5.1 Restarted and truncated versions of GMRES

The full version of GMRES is guaranteed to converge in exact arithmetic in at most n steps, but this may become impractical for the aforementioned requirements in memory storage and computation. The least-squares problem in Equation (2.55) can become challenging if the prescribed accuracy is not reached for small values of k, i.e. in a few iterations. In fact, a considerable growth in k may leads to more demanding requirements both in terms of memory storage and computations. These memory and computational burdens can be alleviated either by adopting restarting or truncating techniques. The restarting approach consists of interrupting the standard GMRES algorithm after a fixed number of iterations and restarting it over from scratch, using the last updated approximate solution as new initial guess. By referring to m as the fixed maximal number of iterations to run before restarting, we denote this variant of GMRES by Restarted GMRES(m).

Algorithm 4: Restarted GMRES(m)

```
  Data: $\mathbf{x}^0$, $tol$
  Result: $\mathbf{x}^m$
1 Compute $\mathbf{r}^0 = \mathbf{b} - A\mathbf{x}^0$;
2 Compute $\beta = \|\mathbf{r}^0\|_2$;
3 Compute $\mathbf{v}^1 = \frac{\mathbf{r}^0}{\beta}$;
4 Generate the Arnoldi basis and the matrix $\overline{H}_m$ using the Arnoldi algorithm
    starting with $\mathbf{v}^1$. Compute $\mathbf{y}_m$, the minimizer of $\|\beta\mathbf{e}_1 - \overline{H}_m\mathbf{y}\|_2$, and
    $\mathbf{x}_m = \mathbf{x}_0 + V_m\mathbf{y}_m$.
5 If satisfied then Stop, else set $\mathbf{x}^0 = \mathbf{x}^m$ and go to 2.
```

A well-known drawback of this restarted variant of GMRES is that it can stagnate and not converge if the matrix is not positive definite. We remind the reader that a real matrix A is *positive definite* if its symmetric part $\frac{1}{2}(A+A^T)$ is symmetric positive definite.

The other attempt to limit the requirements needed is to truncate the Arnoldi basis rather than restarting it after m iterations. A truncated variant of GMRES that adopts this approach is called DQGMRES [54] and it is based on replacing the Arnoldi algorithm with an incomplete orthogonalization as described in Algorithm 5. Only the k previous vectors of the Arnoldi process are kept at every iteration, leading to a banded structure of $\overline{H}_m$.

Algorithm 5: Incomplete Orthogonalization Process

```
  Data: $\mathbf{v}^1$, $A$
  Result: $\{\mathbf{w}_i\}_{i=1}^k$
1 Compute $\mathbf{w}_j = A\mathbf{v}_j$;
2 for $j = \max\{1, j-k+1\}, \ldots, j$ do
3     $h_{ij} = (\mathbf{w}_j, \mathbf{v}_i)$;
4     $\mathbf{w}_j = \mathbf{w}_j - h_{ij}\mathbf{v}_i$
5 end
6 Compute $h_{j+1,j} = \|\mathbf{w}_j\|_2$.
7 Compute $\mathbf{v}_{j+1} = \frac{\mathbf{w}_j}{h_{j+1,j}}$.
```

From now on, we will refer to the full, restarted and truncated variants of GMRES as Full GMRES, Restarted GMRES and Truncated GMRES, respectively.

2.6 Comparison between GMRES and Anderson-Richardson

Both Anderson-Richardson and GMRES aim to identify a projection subspace where to minimize the residual associated with the approximate solution to 2.1. Because of this, these two algorithms may be mathematically equivalent in exact arithmetic under some hypotheses. We present here the study published in [47] as to the connection between Anderson-Richardson and Full-GMRES. First, we introduce some quantities that are employed to describe the performance of the methods. In [62] the *grade* of $\mathbf{r}^0 \neq 0$ with respect to A is defined as

$$\nu(A, \mathbf{r}^0) = \max\{\ell \in \mathbb{N} : \dim(\mathcal{K}_\ell(A, \mathbf{r}^0)) = \ell\}.$$

Alternatively, the grade of $\mathbf{r}^0 \neq 0$ with respect to A can also be equivalently defined as the smallest integer ν for which there is a non-zero monic polynomial q of degree ν such that

$$q(A)\mathbf{r}^0 = 0,$$

i.e.,

$$\nu(A, \mathbf{r}^0) = \min\{\ell \in \mathbb{N} : \ \mathbf{r}^0, A\mathbf{r}^0, \ldots, A^\ell \mathbf{r}^0 \text{ are linearly dependent}\}.$$

The polynomial q is called *minimum polynomial* of $\mathbf{r}^0$ with respect to A. Clearly

$$\nu(A, \mathbf{r}^0) \leq n$$

and

$$\mathcal{K}_\ell(A, \mathbf{r}^0) = \mathcal{K}_\nu(A, \mathbf{r}^0), \quad \forall \ell \geq \nu = \nu(A, \mathbf{r}^0).$$

The following result was initially introduced in [53] and it relates the convergence of Full GMRES to $\nu(A, \mathbf{r}^0)$.

Proposition 3. [53] The Full GMRES method converges in $\nu(A, \mathbf{r}^0)$ steps in exact arithmetic, i.e.,

$$\mathbf{x}_G^k \neq \mathbf{x}, \quad \text{for} \quad k < \nu(A, \mathbf{r}^0), \quad \text{and} \quad \mathbf{x}_G^k = \mathbf{x}, \quad \text{for} \quad k \geq \nu(A, \mathbf{r}^0).$$

Following [47], we introduce the *stagnation index* as

$$\eta_G = \min\{\ell \in \mathbb{N} : \ \mathbf{x}_G^\ell = \mathbf{x}_G^{\ell+1}\}.$$

We note that another way to define the stagnation is used in the literature. It describes stagnation in Full GMRES as the situation where the residual norm does not decrease across consecutive iterations. However, the two definitions are equivalent for Full GMRES.

As concerns AR, we are going to assume that $p = k$ for the rest of this section, which means that every Anderson mixing involves the approximations computed at all the previous iterations. We refer to the method with this particular setting as Full AR. Setting $p = k$ implies that Equation (2.37) becomes

$$\overline{\mathbf{x}}^k_{AR} = \mathbf{x}^k_{AR} - \sum_{i=1}^{k} g^k_i (\mathbf{x}^i_{AR} - \mathbf{x}^{i-1}_{AR}), \tag{2.56}$$

where

$$\mathbf{g}_k = (g^k_1, \ldots, g^k_k)^T = \underset{\mathbf{g}\in\mathbb{R}^k}{\operatorname{argmin}} \left\| \mathbf{b} - A \left[\mathbf{x}^k_{AR} - \sum_{i=1}^{k} g_i (\mathbf{x}^i_{AR} - \mathbf{x}^{i-1}_{AR}) \right] \right\|_2 . \tag{2.57}$$

In Formula (2.57) the vector $\mathbf{x}^i_{AR}$ represents the solution computed with Full AR at the ith iteration. Notice that the size of the least-squares problem (and so the cost of an iteration of Full AR) progressively increases with the iteration index.

From Equation (2.56) we have that $(\overline{\mathbf{x}}^k_{AR} - \mathbf{x}^k_{AR}) \in \mathcal{L}_k$, where

$$\begin{aligned} \mathcal{L}_k &= \operatorname{span}\{(\mathbf{x}^1_{AR} - \mathbf{x}^0), (\mathbf{x}^2_{AR} - \mathbf{x}^1_{AR}), \ldots, (\mathbf{x}^k_{AR} - \mathbf{x}^{k-1}_{AR})\} \\ &= \operatorname{span}\{(\mathbf{x}^1_{AR} - \mathbf{x}^0), (\mathbf{x}^2_{AR} - \mathbf{x}^0), \ldots, (\mathbf{x}^k_{AR} - \mathbf{x}^0)\}. \end{aligned} \tag{2.58}$$

In [47] the authors introduced the *index of Anderson-Richardson* which is defined as

$$\mu_{AR}(A, \mathbf{x}^0) = \min\{\ell \in \mathbb{N} : \ (\mathbf{x}^1_{AR} - \mathbf{x}^0), (\mathbf{x}^2_{AR} - \mathbf{x}^0), \ldots, (\mathbf{x}^\ell_{AR} - \mathbf{x}^0) \text{ are linearly dependent}\}.$$

Notice that the linear dependence of the set of vectors

$$(\mathbf{x}^1_{AR} - \mathbf{x}^0), (\mathbf{x}^2_{AR} - \mathbf{x}^0), \ldots, (\mathbf{x}^\ell_{AR} - \mathbf{x}^0)$$

is equivalent to the linear dependence of the vectors

$$(\mathbf{x}_{AR}^1 - \mathbf{x}^0), (\mathbf{x}_{AR}^2 - \mathbf{x}_{AR}^1), \dots, (\mathbf{x}_{AR}^\ell - \mathbf{x}_{AR}^{\ell-1}).$$

The index $\mu_{AR}(A, \mathbf{x}^0)$ is useful to describe the convergence properties of Full AR. In fact, $\mu_{AR}(A, \mathbf{x}^0)$ can be used in the following result that establishes a close relation between Full GMRES and Full AR. The result was first introduced in [61] assuming that all the relaxation parameters were set to 1. The result was extended in [47] to consider also the case of general values for the relaxation parameters β_k's. We include also the proof since some steps of it are useful for future discussions.

Proposition 4. [47] The index $\mu_{AR}(A, \mathbf{x}^0)$ of Anderson-Richardson is always less than or equal to the grade $\nu(A, \mathbf{r}^0)$. Moreover, the sequence of solutions $\{\mathbf{x}_G^k\}$ computed via Full GMRES and the sequence of solutions $\{\mathbf{x}_{AR}^k\}$ computed via Full AR satisfy the following properties:

- $\mathbf{x}_{AR}^{k+1} = \mathbf{x}_G^k + \beta_k(\mathbf{b} - A\mathbf{x}_G^k)$, $k = 0, 1, \dots, \mu_{AR}(A, \mathbf{x}^0)$;
- $\overline{\mathbf{x}}_{AR}^k = \mathbf{x}_G^k$, $k = 0, 1, \dots, \mu_{AR}(A, \mathbf{x}^0)$.

Proof. Let us first prove by induction that $\mathcal{L}_k \subseteq \mathcal{K}_k(A, \mathbf{r}^0)$. The base case for $k = 1$ is true because $\mathbf{x}_{AR}^1 - \mathbf{x}^0 = \beta_0 \mathbf{r}^0$. Therefore, now we can focus on the induction step. By assuming that $\mathcal{L}_k \subseteq \mathcal{K}_k(A, \mathbf{r}^0)$ we aim to prove that this induces $\mathcal{L}_{k+1} \subseteq \mathcal{K}_{k+1}(A, \mathbf{r}^0)$. From the definition of the Anderson-Richardson iteration in Equations (2.44a) and (2.44b) together with the alternate formulation of Anderson mixing in Equation (2.41) it follows that

$$\begin{aligned}
\mathbf{x}_{AR}^{k+1} &= \overline{\mathbf{x}}_{AR}^{k} + \beta_k(\mathbf{b} - A\overline{\mathbf{x}}_{AR}^{k}) \\
&= \left[\mathbf{x}_{AR}^{k} - \sum_{i=1}^{k} g_i^k(\mathbf{x}_{AR}^{i} - \mathbf{x}_{AR}^{i-1})\right] + \beta_k\left[\mathbf{b} - A\left(\mathbf{x}_{AR}^{k} - \sum_{i=1}^{k} g_i^k(\mathbf{x}_{AR}^{i} - \mathbf{x}_{AR}^{i-1})\right)\right] \\
&= \left[\mathbf{x}^0 - \sum_{i=1}^{k} \hat{g}_i^k(\mathbf{x}_{AR}^{i} - \mathbf{x}^0)\right] + \beta_k\left[\mathbf{b} - A\left(\mathbf{x}^0 - \sum_{i=1}^{k} \hat{g}_i^k(\mathbf{x}_{AR}^{i} - \mathbf{x}^0)\right)\right] \\
&= \mathbf{x}^0 + \beta_k\mathbf{r}^0 - (I - \beta_k A)\sum_{i=1}^{k} \hat{g}_i^k(\mathbf{x}_{AR}^{i} - \mathbf{x}^0).
\end{aligned}$$

Therefore,

$$\mathbf{x}_{AR}^{k+1} \in \mathcal{L}_{k+1} = \mathcal{L}_k + A\mathcal{L}_k \subseteq \mathcal{K}_k(A, \mathbf{r}^0) + A\mathcal{K}_k(A, \mathbf{r}^0) = \mathcal{K}_{k+1}(A, \mathbf{r}^0),$$

which completes the induction step.

Since

$$\mathcal{L}_k \subseteq \mathcal{K}_k(A, \mathbf{r}^0), \quad k \leq \mu_{AR}(A, \mathbf{x}^0),$$

the linear independence of $(\mathbf{x}_{AR}^1 - \mathbf{x}^0), \ldots, (\mathbf{x}_{AR}^k - \mathbf{x}^0)$ always implies the linear independence of $\mathbf{r}^0, \ldots, A^{k-1}\mathbf{r}^0$. Therefore,

$$\mu_{AR}(A, \mathbf{x}^0) \leq \nu(A, \mathbf{r}^0),$$

which leads to

$$\mathcal{L}_k = \mathcal{K}_k(A, \mathbf{r}^0), \quad \text{and} \quad dim(\mathcal{K}_k(A, \mathbf{r}^0)) = k, \quad \text{for} \quad k = 1, \ldots, \mu_{AR}(A, \mathbf{x}^0).$$

The second bullet of the statement is due to the fact that both $\overline{\mathbf{x}}_{AR}^k$ and $\mathbf{x}_G^k$ are computed to minimize the residual on the same Euclidean subspace $\mathcal{K}_k(A, \mathbf{r}^0)$. Then the first bullet follows from Equation (2.44b). $\square$

Therefore, Full AR converges to the solution $\mathbf{x}$ of the linear system (2.1) if

$$\mu_{AR}(A, \mathbf{x}^0) = \nu(A, \mathbf{r}^0).$$

Otherwise the condition $\mu_{AR}(A, \mathbf{x}^0) < \nu(A, \mathbf{r}^0)$ implies that $\mu_{AR}(A, \mathbf{x}^0)$ is precisely the first index for which Full GMRES stagnates, meaning that $\mu_{AR}(A, \mathbf{x}^0) = \eta_G$. When a stagnation occurs, Full GMRES continues to iteratively augment the Krylov subspace and it eventually converges to the solution $\mathbf{x}$ in exact arithmetic, while Full AR stagnates from that point on without ever recovering. This can be explained through the sequence of subspaces generated by Full AR and Full GMRES. If Full AR and Full GMRES stagnate between the iterations with index k and $k+1$, then we obtain

$$\overline{\mathbf{x}}_{AR}^{k} = \overline{\mathbf{x}}_{AR}^{k+1} = \mathbf{x}_G^k = \mathbf{x}_G^{k+1}.$$

This can only occur if $(\mathbf{x}_{AR}^{k+1} - \mathbf{x}^0) \in \mathcal{L}_k$. Therefore,

$$\mathcal{L}_{k+1} = \mathcal{L}_k \neq \mathcal{K}_{k+1}(A, \mathbf{r}^0),$$

meaning that the equivalence between Full AR and Full GMRES holds only in the absence of stagnation.

By comparing the basis generated by Full GMRES and Full AR at each iteration, one can see that the two bases are different. The only term in common is the first one (up to a scaling factor), since this vector is represented by the initial residual $\mathbf{r}^0$. In fact, the first normalized vector $\mathbf{v}_1$ of the basis generated by Full GMRES is

$$\mathbf{v}_1 = \frac{\mathbf{r}^0}{\|\mathbf{r}^0\|_2}$$

whereas the first vector $\mathbf{x}_{AR}^1 - \mathbf{x}^0$ of the basis generated by Full AR is

$$\mathbf{x}_{AR}^1 - \mathbf{x}^0 = \beta_0 \mathbf{r}^0.$$

We provide now an explicit expression of the second term of the bases for Full GMRES and Full AR. The non-normalized second vector $\mathbf{w}_1$ of the basis constructed by Full GMRES is

$$\begin{aligned}
\mathbf{w}_1 &= A\mathbf{v}_1 - (A\mathbf{v}_1, \mathbf{v}_1)\mathbf{v}_1 \\
&= A\mathbf{v}_1 - (\mathbf{v}_1^T A\mathbf{v}_1)\mathbf{v}_1 \\
&= (I - \mathbf{v}_1\mathbf{v}_1^T)A\mathbf{v}_1 \\
&= \left[I - \frac{\mathbf{r}^0}{\|\mathbf{r}^0\|_2} \left(\frac{\mathbf{r}^0}{\|\mathbf{r}^0\|_2} \right)^T \right] \frac{A\mathbf{r}^0}{\|\mathbf{r}^0\|_2}.
\end{aligned}$$

With regards to Full AR, we know that the Anderson correction can be defined in two different ways, as shown in Equations (2.37) and (2.41). If one adopts the formula in Equation (2.37), then the second term of the basis constructed by Full AR is $\mathbf{x}_{AR}^2 - \mathbf{x}_{AR}^1$. Otherwise the formula in Equation (2.41) leads to expanding the basis with the vector $\mathbf{x}_{AR}^2 - \mathbf{x}^0$. Here we are going to provide explicit representations of both $\mathbf{x}_{AR}^2 - \mathbf{x}_{AR}^1$ and $\mathbf{x}_{AR}^2 - \mathbf{x}^0$ in terms of $\mathbf{r}^0$. This allows us to directly compare the expressions of these vectors with the one of $\mathbf{w}_1$ and understand whether they identify the same direction or not. We temporarily set all the relaxation parameters β_k's to 1 for the sake of simplicity. The first iteration of Full AR is accomplished the same way independently of the specific approach adopted to implement the method, obtaining (see (2.38)-(2.39))

$$X_1 = \hat{X}_1 = (\mathbf{x}_{AR}^1 - \mathbf{x}^0) = \mathbf{r}^0$$

and

$$R_1 = \hat{R}_1 = (\mathbf{r}_{AR}^1 - \mathbf{r}^0) = -AX_1 = -A\hat{X}_1 = -A\mathbf{r}^0,$$

where we used the fact that $\mathbf{r}^1_{AR} = (I - A)\mathbf{r}^0$. Once $\mathbf{x}^1_{AR}$ is computed, it is required to apply an Anderson acceleration. This produces an updated approximation $\overline{\mathbf{x}}^1_{AR}$ such that

$$\overline{\mathbf{x}}^1_{AR} = \mathbf{x}^0 + \frac{(\mathbf{r}^0)^T A\mathbf{r}^0}{\|A\mathbf{r}^0\|_2^2}\mathbf{r}^0$$

and the approximation obtained by Full AR at the end of the second iteration is

$$\begin{aligned}
\mathbf{x}^2_{AR} &= \overline{\mathbf{x}}^1_{AR} + (\mathbf{b} - A\overline{\mathbf{x}}^1_{AR}) \\
&= \mathbf{x}^0 + \frac{(\mathbf{r}^0)^T A\mathbf{r}^0}{\|A\mathbf{r}^0\|_2^2}\mathbf{r}^0 + \left[\mathbf{b} - A\left(\mathbf{x}^0 + \frac{(\mathbf{r}^0)^T A\mathbf{r}^0}{\|A\mathbf{r}^0\|_2^2}\mathbf{r}^0\right)\right] \\
&= \mathbf{x}^0 + \frac{(\mathbf{r}^0)^T A\mathbf{r}^0}{\|A\mathbf{r}^0\|_2^2}\mathbf{r}^0 + \mathbf{r}^0 - \frac{(\mathbf{r}^0)^T A\mathbf{r}^0}{\|A\mathbf{r}^0\|_2^2}A\mathbf{r}^0.
\end{aligned}$$

Full AR proceeds up to this point indistinctly with respect to the use of formula (2.37) rather than (2.41). Now, if Full AR were implemented according to the formula in (2.37), the second term added to the basis would be the vector

$$\mathbf{x}^2_{AR} - \mathbf{x}^1_{AR} = \frac{(\mathbf{r}^0)^T A\mathbf{r}^0}{\|A\mathbf{r}^0\|_2^2}(I - A)\mathbf{r}^0.$$

The vector $\mathbf{x}^2_{AR} - \mathbf{x}^1_{AR}$ is used to build X_2 based on the definition in (2.38). On the other hand Full AR implemented through the formula in (2.41) would expand the basis with

$$\mathbf{x}^2_{AR} - \mathbf{x}^0 = \mathbf{r}^0 + \frac{(\mathbf{r}^0)^T A\mathbf{r}^0}{\|A\mathbf{r}^0\|_2^2}(I - A)\mathbf{r}^0.$$

The vector $\mathbf{x}^2_{AR} - \mathbf{x}^0$ is used to build $\hat{X}_2$ according to the definition in (2.42). This shows that in general $\mathbf{w}_1$, $\mathbf{x}^2_{AR} - \mathbf{x}^1_{AR}$ and $\mathbf{x}^2_{AR} - \mathbf{x}^0$ are not collinear, since a scalar multiplying factor is not enough to reconstruct one of these vectors from any of the others. The absence of collinearity among the vectors used to expand the bases propagates across all the successive iterations as well, causing

$$\begin{aligned}
&\operatorname{span}\{\mathbf{w}_k\} \neq \operatorname{span}\{\mathbf{x}_{AR}^k - \mathbf{x}^0\},\\
&\operatorname{span}\{\mathbf{w}_k\} \neq \operatorname{span}\{\mathbf{x}_{AR}^k - \mathbf{x}_{AR}^{k-1}\},\\
&\operatorname{span}\{\mathbf{x}_{AR}^k - \mathbf{x}^0\} \neq \operatorname{span}\{\mathbf{x}_{AR}^k - \mathbf{x}_{AR}^{k-1}\}, \quad k \geq 2.
\end{aligned}$$

Therefore, the bases generated by the two implementations of Full AR and by Full GMRES are all different. Nevertheless, they still identify the same subspace as long as the bases are updated with new vectors that preserve linear independence. In particular, Full AR and Full GMRES are additive oblique projections methods that are mathematically equivalent in absence of stagnation because

$$\mathcal{V}_k = \mathcal{K}_k(A, \mathbf{r}^0) = \mathcal{R}(X_k) = \mathcal{R}(\hat{X}_k), \quad \dim(\mathcal{V}_k) = p = k$$

and

$$\mathcal{W}_k = A\mathcal{K}_k(A, \mathbf{r}^0) = A\mathcal{R}(X_k) = A\mathcal{R}(\hat{X}_k) = \mathcal{R}(R_k) = \mathcal{R}(\hat{R}_k), \quad \dim(\mathcal{W}_k) = p = k.$$

In the case of p fixed across all the iterations, the columns of the matrix X_k and R_k (or $\hat{X}_k$ and $\hat{R}_k$ depending on which implementation of AR is considered) are iteratively updated by dropping out old vectors and including new ones. This makes AR comparable with Truncated GMRES [54]. However, the algorithms differ in the subspace onto which the linear system in (2.1) is projected. To explain this, let us denote with $\{\mathbf{v}_i\}_{i=1}^k$ the basis generated by the Arnoldi process in the first k iterations of GMRES

$$\mathcal{K}_k(A, \mathbf{r}^0) = \operatorname{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_k\},$$

and let us recall the definition of $\mathcal{L}_k$ in (2.58). We already showed that the vectors used by AR and GMRES to expand the bases do not coincide. In general a truncation of the bases leads to different subspaces:

$$\begin{aligned}
&\text{span}\{\mathbf{v}_\ell, \dots, \mathbf{v}_k\} \neq \text{span}\{(\mathbf{x}^\ell_{AR} - \mathbf{x}^{\ell-1}_{AR}), \dots, (\mathbf{x}^k_{AR} - \mathbf{x}^{k-1}_{AR})\}\\
&\text{span}\{\mathbf{v}_\ell, \dots, \mathbf{v}_k\} \neq \text{span}\{(\mathbf{x}^\ell_{AR} - \mathbf{x}^0), \dots, (\mathbf{x}^k_{AR} - \mathbf{x}^0)\},\\
&\text{span}\{(\mathbf{x}^\ell_{AR} - \mathbf{x}^{\ell-1}_{AR}), \dots, (\mathbf{x}^k_{AR} - \mathbf{x}^{k-1}_{AR})\} \neq \text{span}\{(\mathbf{x}^\ell_{AR} - \mathbf{x}^0), \dots, (\mathbf{x}^k_{AR} - \mathbf{x}^0)\},\\
&1 < \ell \leq k.
\end{aligned}$$

This implies that Truncated GMRES and AR with a fixed value of p (we will use the term Truncated AR from now on) use different constraints to build the projection subspace. Therefore, the performances of Truncated GMRES and Truncated AR do not generally match. This will be validated also by numerical examples provided in Section 2.7.1. Furthermore, we stress that the two implementations of Truncated AR also differ between each other, since the two truncated bases do not identify the same projection subspace.

In case the matrices X_k and R_k (or $\hat{x}_k$ and $\hat{R}_k$) used by Anderson-Richardson are flushed after every batch of m iterations and the procedure is restarted with the updated residual, the performance of Anderson-Richardson is similar to Restarted GMRES(m) [52, 61]. We call Restarted AR(m) this version of Anderson-Richardson. Although Restarted GMRES(m) and Restarted AR(m) are very similar, they are not equivalent. In fact, Restarted AR(m) employs a least-squares problem followed by a Richardson step at each iteration, as shown in Equations (2.44a)-(2.44b). This entails that Restarted AR(m) and Restarted GMRES(m) would coincide only if the last Richardson sweep at the end of each cycle of Restarted AR(m) were discarded. This disparity between Restarted AR(m) and Restarted GMRES(m) may sometimes favor the former over the latter. In fact, the last Richardson sweep in every cycle of Restarted AR(m) may allow Restarted AR(m) to reach convergence in those situations where Restarted GMRES(m) is known to stagnate. This fact will be confirmed by numerical experiments in Section 2.7.2. This discussion disproves statements in [41,61], where the authors state the mathematical equivalence between truncated and restarted versions of GMRES and AR.

2.6.1 Convergence Analysis for Anderson-Richardson

The mismatch between Full AR and Full GMRES when $\mu_{AR} < \nu(A, \mathbf{r}^0)$ highlights the need to come up with convergence criteria for Anderson-Richardson, since the theory developed for Full GMRES cannot be entirely reused to this end. Several contributions in the literature considered the use of Anderson acceleration for non-linear problems, but less attention has been paid to the linear case. One of the works dealing with the linear case is [60]. In this work the authors proved that Anderson-Richardson is guaranteed to converge to the solution of (2.1) if the iteration matrix $H = I - A$ is such that $\|H\|_2 < 1$. We include here below the proof of this theorem because it will facilitate future discussions. To do so, let us recall that a sequence $\{\mathbf{w}^k\}$ converges q-linearly with q-factor $c \in [0, 1)$ to $\mathbf{w}^*$ if

$$\|\mathbf{w}^{k+1} - \mathbf{w}^*\| \leq c\|\mathbf{w}^k - \mathbf{w}^*\|,$$

for all $k \geq 1$. We are going to assume that all the relaxation parameters β_k are set to 1 for the rest of this section. In fact, if $\|H\|_2 < 1$, the underlying Richardson iteration is already convergent without the use of relaxation parameters.

Theorem 2.6.1. *Consider a linear system as in Equation* (2.1)*. If the iteration matrix $H = I - A$ is such that $\|H\|_2 = c < 1$, then Anderson-Richardson converges to the solution $\mathbf{x}$ of* (2.1) *for any choice of the parameter p, regardless of the initial guess. The residual norm converges to zero q-linearly and the q-factor is c.*

Proof. We recall the definition of $\overline{\mathbf{x}}_{AR}^k$ as in Equation (2.37) or (2.41) for a generic value of p. We have

$$\mathbf{x}_{AR}^{k+1} = \overline{\mathbf{x}}_{AR}^{k} + \overline{\mathbf{r}}_{AR}^{k}$$

and

$$\mathbf{r}_{AR}^{k+1} = H\overline{\mathbf{r}}_{AR}^{k},$$

where $\overline{\mathbf{r}}_{AR}^{k} = \mathbf{b} - A\overline{\mathbf{x}}_{AR}^{k}$ and $\mathbf{r}_{AR}^{k+1} = \mathbf{b} - A\mathbf{x}_{AR}^{k+1}$. Therefore,

$$\|\mathbf{r}_{AR}^{k+1}\|_2 \leq \|H\|_2 \|\overline{\mathbf{r}}_{AR}^{k}\|_2 \leq c\|\overline{\mathbf{r}}_{AR}^{k}\|_2.$$

The Anderson acceleration computes a linear mixing to minimize the residual. Therefore, the Anderson acceleration computes an update $\overline{\mathbf{x}}_{AR}^{k+1}$ whose residual

$$\overline{\mathbf{r}}_{AR}^{k+1} = \mathbf{b} - A\overline{\mathbf{x}}_{AR}^{k+1}$$

has to satisfy the following chain of inequalities:

$$\|\overline{\mathbf{r}}_{AR}^{k+1}\|_2 \leq \|\mathbf{r}_{AR}^{k+1}\|_2 \leq c\|\overline{\mathbf{r}}_{AR}^{k}\|_2.$$

The last chain of equalities must hold regardless of the value of p. This proves that the residual ℓ^2-norm converges to zero q-linearly and the q-factor is c. □

This result guarantees convergence of Anderson-Richardson only under the restrictive assumption that $\|H\|_2 < 1$. Relaxing the requirement to $\rho(H) < 1$ already gives room to non-normal iteration matrices that could hinder convergence. In general the parameter p is fixed to prevent the cost of the least-squares problem from increasing.

2.7 Numerical experiments

This section of numerical experiments aims to support the theoretical discussion conducted in the previous sections of this chapter.

2.7.1 Comparison between Truncated AR and Truncated GMRES

In Section 2.6 we argued why Truncated GMRES and Truncated AR are not mathematically equivalent. We provide here some numerical experiments that validate this fact. The matrices we consider are all taken from the SuiteSparse Matrix Collection [18]. The linear systems have been solved with both Truncated GMRES and Truncated AR, setting the truncation parameter to 10. Therefore, only the last 10 terms of the basis are kept at each iteration. The linear systems are preconditioned via Incomplete LU factorization with zero fill-in in the factors [52] (shortly referred as ILU(0) from now on) and a relative residual less than 10^{-8} is used as stopping criterion. The solution to the linear system is generated randomly and the right hand side is obtained multiplying the solution vector by the coefficient matrix A. The same linear system is used for both the algorithms, so that the comparison between performances is legitimate. The results are shown in Table 2.1. It is noticed that the numbers of iterations employed by the algorithms are significantly different. This provides numerical support to the theoretical claim about Truncated GMRES and Truncated AR not being mathematically equivalent. We also show the residual trend for both algorithms applied to a linear system with the matrix `nos3` in Figure 2.1. The algorithms behave identically for the first 10 iterations, since no truncation has occurred yet and the subspaces constructed by Truncated AR and Truncated GMRES coincide. However, the residual curves start departing from each other when the truncation of the bases takes place, since the projection subspaces differ.

		Truncated AR$(p = 10)$		**Truncated GMRES**$(m = 10)$	
Matrix	**size**	rel. error	# Itr.	rel. error	# Itr.
chipcool0	20,082	$1.70 \cdot 10^{-7}$	177	$2.80 \cdot 10^{-7}$	87
memplus	17,758	$6.60 \cdot 10^{-6}$	426	$2.30 \cdot 10^{-5}$	305
nos3	960	$9.05 \cdot 10^{-7}$	194	$9.01 \cdot 10^{-7}$	245
saylr4	3,564	$9.03 \cdot 10^{-8}$	294	$9.10 \cdot 10^{-9}$	447
sherman3	5,005	$4.50 \cdot 10^{-6}$	203	$7.30 \cdot 10^{-6}$	246
sherman5	3,312	$3.96 \cdot 10^{-7}$	49	$1.20 \cdot 10^{-7}$	63

Table 2.1: Comparison between Truncated AR and Truncated GMRES. Experiments with IC(0) preconditioner for symmetric positive definite matrices or ILU(0) preconditioner that are not symmetric positive definite.

2.7.2 Comparison between Restarted AR and Restarted GMRES

At the end of Section 2.6 we mentioned the fact that Restarted AR(m) is not exactly equivalent to Restarted GMRES(m), since an AR iteration consists of a least-squares problem followed by a standard Richardson sweep. As already mentioned earlier, the presence of the last Richardson sweep at the end of each cycle can favor Restarted AR(m) over Restarted GMRES(m) in those situations where the latter stagnates without any later recovery. This happens for examples taken from [23], where the author demonstrates that an increase of the restarting parameter m does not necessarily improve the performance of Restarted GMRES(m). The examples used in [23] are the following two 3×3 linear systems $A_i \mathbf{x}_i = \mathbf{b}_i$:

$$A_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{b}_1 = \begin{bmatrix} 2 \\ -4 \\ 1 \end{bmatrix}, \tag{2.59}$$

$$A_2 = \begin{bmatrix} 1 & 2 & -2 \\ 0 & 2 & 4 \\ 0 & 0 & 3 \end{bmatrix}, \quad \mathbf{b}_2 = \begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix}. \tag{2.60}$$

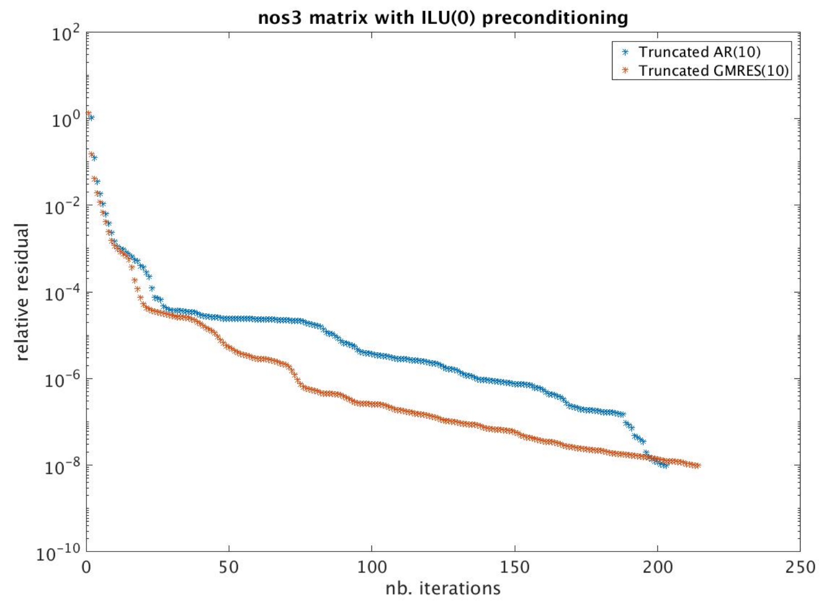

Figure 2.1: Truncated GMRES vs. Truncated AR. Plot of the history of relative residual ℓ^2-norm for the matrix `nos3` with IC(0) preconditioner.

One can verify that GMRES(2) stagnates on both these linear system with a final relative residual of approximately 0.60146 and 0.29329, respectively. However, Restarted AAR(2) converges in three iterations for Problem (2.59) and four iterations for Problem (2.60), driving the relative residual down to machine precision. This is due to the stagnation-recovery property of the final Richardson step.

Chapter 3

Alternating Anderson-Richardson

3.1 Introduction

Many applications related to quantitative disciplines such as engineering, physics and finance challenge state-of-the-art computing architectures requiring heavier computations that must be performed in a timely manner. Upcoming computers are projected to meet this requirements by increasing the number of computational nodes composing the processors grid and thus enhancing the theoretical peak of floating point operations per second. This increase of computational nodes in a platform causes a significant upsurge of the cost of global communications with respect to older generation computers. Standard Krylov methods presented in Chapter 2 do not address these new computational issues, since they are expected to incur in expensive communication overheads on new computers moving towards exsscale capacities. For instance, CG and GMRES perform inner products at each iteration, whereas AR solves an unstructured least-squares problem at each iteration. All these tasks require global communications across all the processes, which causes severe bottlenecks for the parallelization.

Therefore, the computer manufacturing trend urges the development of linear solvers that cap the amount of global communication needed to accomplish the computational tasks. In this Chapter we focus on the analysis and development of linear solvers targeted to reduce the amount of communication needed to solve a linear system in a parallel computing framework. A recently proposed algorithm that tackles this issue is the *Alternating Anderson-Richardson* method [48, 49].

3.2 Alternating Anderson-Richardson method

The Alternating Anderson-Richardson method (AAR for short) computes an Anderson mixing after multiple Richardson steps, so that the cost of solving successive least-squares problems is mitigated by several cheap Richardson sweeps in between. A pseudo-code of the algorithm is provided in Algorithm 6. We present the formulation of AAR assuming that the Anderson acceleration is implemented based on Equation (2.37). The iteration of the AAR method has the form of a non-stationary Richardson scheme

$$\mathbf{x}^{k+1} = \mathbf{x}^k + C_k \mathbf{r}^k,$$

where now the matrix C_k becomes

$$C_k = \begin{cases} \omega I, & k/m \notin \mathbb{N} \\ \beta_k I - (X_k + \beta_k R_k)(R_k^T R_K)^{-1} R_k^T, & k/m \in \mathbb{N} \end{cases} .$$

The matrices X_k and R_k are defined as in (2.38) and (2.39), respectively. The parameter m represents the number of Richardson sweeps separating two consecutive Anderson mixing accelerations. The other parameters are ω, $\{\beta_k\}$ and p. The first two are relaxation parameters for the Richardson sweeps, whereas p represents the number of columns in the $n \times p$ least-squares problem to compute the Anderson acceleration. It should be noted that the least-squares problems used in this scheme are not structured. This is a disadvantage in comparison with other standard linear solvers like GMRES, where the least-squares problems are cheaper to solve because the matrices are structured as upper Hessenberg matrices. Letting $m \to \infty$ would reduce AAR to stationary Richardson with ω as relaxation parameter, while $m = 1$ would make it coincide with Anderson-Richardson where $\{\beta_k\}$ would be the sequence of relaxation parameters. We stress the fact that an alternative would be to adopt the definition of Anderson correction as in Equation (2.41) along with the definition of matrices $\hat{X}_k$ and $\hat{R}_k$ as in (2.42) and (2.43). However, this implementation of the Anderson acceleration does not allow us to recast the iteration of AAR so that $\mathbf{x}^{k+1}$ directly relates to $\mathbf{x}^k$ whenever k is a multiple of m.

Algorithm 6: Alternating Anderson-Richardson (AAR)

```
   Data: $\mathbf{x}^0$, $\omega$, $\{\beta_k\}_{k=0}^{k=\text{maxit}}$, $p$
   Result: $\mathbf{x}^{k+1}$
1  Compute $\mathbf{r}^0 = \mathbf{b} - A\mathbf{x}^0$;
2  Compute $\mathbf{x}^1 = (I - \beta_0 A)\mathbf{x}^0 + \mathbf{b}$;
3  Set $k = 1$;
4  while $\frac{\|\mathbf{r}^{k-1}\|_2}{\|\mathbf{b}\|_2} > tol$ do
5      Compute $\ell = \min\{k, p\}$;
6      Compute $\mathbf{r}^k = \mathbf{b} - A\mathbf{x}^k$;
7      Set $X_k = [(\mathbf{x}^{k-\ell+1} - \mathbf{x}^{k-\ell}), \dots, (\mathbf{x}^k - \mathbf{x}^{k-1})] \in \mathbb{R}^{n\times\ell}$;
8      Set $R_k = [(\mathbf{r}^{k-\ell+1} - \mathbf{r}^{k-\ell}), \dots, (\mathbf{r}^k - \mathbf{r}^{k-1})] \in \mathbb{R}^{n\times\ell}$;
9      if $k \,\%\, m \neq 0$ then
10         Set $\mathbf{x}^{k+1} = \mathbf{x}^k + \omega\mathbf{r}^k$;
11     end
12     else
13         Determine $\mathbf{g}^k = [g_1^k, \dots, g_\ell^k]^T$ such that $\mathbf{g}^k = \underset{\mathbf{g}\in\mathbb{R}^\ell}{\text{argmin}}\|\mathbf{r}^k - R_k\mathbf{g}\|_2$;
14         Set $\overline{\mathbf{x}}^k = \mathbf{x}^k - X_k\mathbf{g}^k$;
15         Set $\mathbf{x}^{k+1} = \overline{\mathbf{x}}^k + \beta_k(\mathbf{b} - A\overline{\mathbf{x}}^k)$;
16     end
17     $k = k + 1$.
18 end
```

3.2.1 Comparison between GMRES and Alternating Anderson-Richardson

Similarly to the work in [47,61] on AR, it is possible to identify a connection between Alternating Anderson-Richardson and GMRES [52]. Let us assume again that $p = k$ at first. Hence, the projection subspace for the Anderson mixing is constructed by using all the previous approximations. From now on we refer to AAR with $p = k$ as Full AAR. We establish the following new result regarding the behavior of this method.

Theorem 3.2.1. *Consider Full AAR with a periodic updating interval equal to m between two consecutive Anderson mixing steps. Denote $\overline{\mathbf{x}}^{\ell m}_{AAR}$ the approximate solution to* (2.1) *provided by Full AAR after the Anderson mixing at the (ℓm)th iteration and denote $\mathbf{x}^{\ell m}_{G}$ the solution computed via Full GMRES at the (ℓm)th iteration. Refer to $\nu(A, \mathbf{r}^0)$ as the grade of $\mathbf{r}^0$ with respect to A and η_G as the stagnation index of Full GMRES. Then the following relation holds in exact arithmetic:*

$$\overline{\mathbf{x}}^{\ell m}_{AAR} = \mathbf{x}^{\ell m}_{G}, \quad \ell = 0, 1, \ldots \quad \text{s.t.} \quad \ell m \leq \eta_G \quad \text{or} \quad \text{s.t.} \quad \ell m \leq \nu(A, \mathbf{r}^0). \tag{3.1}$$

Moreover, if the stagnation does not occur, then

$$\overline{\mathbf{x}}^{tm}_{AAR} = \mathbf{x}^{\nu(A,\mathbf{r}^0)}_{G} = \mathbf{x}, \quad t \in \mathbb{N}, \quad \text{s.t.} \quad (t-1)m < \nu(A, \mathbf{r}^0) \leq tm. \tag{3.2}$$

Proof. Without loss of generality we set $\omega = 1$ and $\beta_k = 1$, $\forall k \geq 0$. Moreover, we remind the reader of the following relation:

$$\eta_G < \nu(A, \mathbf{r}^0).$$

Indeed, if the iteration index hit $\nu(A, \mathbf{r}^0)$, then Full GMRES would converge to the exact solution without stagnating. We start considering the first m Richardson sweeps and we express the approximations to the solution of (2.1) in terms of powers of A that multiply the initial residual $\mathbf{r}^0 = \mathbf{b} - A\mathbf{x}^0$. Our initial claim is that

$$\overline{\mathbf{x}}_{AAR}^{m} = \mathbf{x}_G^m. \tag{3.3}$$

The proof of this initial claim proceeds in an inductive fashion. The first Richardson iteration yields

$$\mathbf{x}_{AAR}^{1} = \mathbf{x}^0 + \mathbf{r}^0,$$

which proves the base case. For the induction step, we assume that

$$\mathbf{x}_{AAR}^{k} = \mathbf{x}^0 + \sum_{i=0}^{k-1} c_i A^i \mathbf{r}^0, \quad k \leq m-1.$$

The successive Richardson step generates a new approximation

$$\begin{aligned}
\mathbf{x}_{AAR}^{k+1} &= \mathbf{x}_{AAR}^{k} + \mathbf{r}^k \\
&= \mathbf{x}^0 + \sum_{i=0}^{k-1} c_i A^i \mathbf{r}^0 + \mathbf{b} - A\Big(\mathbf{x}^0 + \sum_{i=0}^{k-1} c_i A^i \mathbf{r}^0\Big) \\
&= \mathbf{x}^0 + \mathbf{r}^0 + \Big(\sum_{i=0}^{k-1} c_i A^i \mathbf{r}^0\Big) - A\Big(\sum_{i=0}^{k-1} c_i A^i \mathbf{r}^0\Big) \\
&= \mathbf{x}^0 + \mathbf{r}^0 + (I - A)\Big(\sum_{i=0}^{k-1} c_i A^i \mathbf{r}^0\Big).
\end{aligned}$$

This leads to

$$\mathbf{x}_{AAR}^{k+1} = \mathbf{x}^0 + \mathbf{z}^{k+1}, \quad k \leq m-1,$$

where $\mathbf{z}^{k+1} \in \mathcal{K}_{k+1}(A, \mathbf{r}^0)$, with $\mathcal{K}_m(A, \mathbf{r}^0)$ being the m-dimensional Krylov subspace defined on matrix A and vector $\mathbf{r}^0$. After the first m Richardson sweeps, the algorithm performs an Anderson mixing as defined in Equation (2.41) yielding

$$\overline{\mathbf{x}}_{AAR}^{m} = \mathbf{x}^0 - \sum_{i=1}^{m} \hat{g}_i^m (\mathbf{x}_{AAR}^i - \mathbf{x}^0),$$

where

$$\hat{\mathbf{g}}^m = (\hat{g}_1^m, \ldots, \hat{g}_m^m)^T = \underset{\mathbf{g}\in\mathbb{R}^m}{\operatorname{argmin}} \left\| \mathbf{b} - A\left[\mathbf{x}^0 - \sum_{i=1}^{m} g_i (\mathbf{x}_{AAR}^i - \mathbf{x}^0)\right] \right\|_2 .$$

The columns of the matrix $\hat{X}_k$ (defined in Equation (2.42)) are expressed in terms of increasing powers of A. Since we are assuming that $\ell m < \nu(A, \mathbf{r}^0)$, this guarantees that matrices $\hat{X}_k$ and $\hat{R}_k$ (defined in Equation (2.43)) have full column rank and

$$\mathcal{R}(\hat{X}_k) = \mathcal{K}_k(A, \mathbf{r}^0), \quad k \leq m.$$

Since the least-squares problem minimizes the residual on the mth Krylov subspace $\mathcal{K}_m(A, \mathbf{r}^0)$, we have proved that

$$\overline{\mathbf{x}}_{AAR}^{m} = \mathbf{x}_G^m.$$

This completes the induction proof for the first m iterations of AAR. Repeating the same induction procedure for successive Richardson and Anderson mixing steps, one can prove the original statement in Formula (3.1). In fact, let us assume that the Anderson mixing at the $(\ell-1)m$th iteration computes

$$\overline{\mathbf{x}}_{AAR}^{(\ell-1)m} = \mathbf{x}^0 + \overline{\mathbf{z}}^{(\ell-1)m}, \quad \overline{\mathbf{z}}^{(\ell-1)m} \in \mathcal{K}_{(\ell-1)m}(A, \mathbf{r}^0)$$

and that $\hat{X}_{(\ell-1)m}$ and $\hat{R}_{(\ell-1)m}$ have full column rank. Then

$$\mathcal{R}(\hat{X}_{(\ell-1)m}) = \mathcal{K}_{(\ell-1)m}(A, \mathbf{r}^0).$$

The approximation $\mathbf{x}_{AAR}^{(\ell-1)m+1}$ is computed via a standard Richardson sweep,

$$\begin{aligned}\mathbf{x}_{AAR}^{(\ell-1)m+1} &= \overline{\mathbf{x}}_{AAR}^{(\ell-1)m} + \mathbf{b} - A\overline{\mathbf{x}}_{AAR}^{(\ell-1)m} \\ &= \mathbf{x}^0 + \overline{\mathbf{z}}^{(\ell-1)m} + \mathbf{b} - A(\mathbf{x}^0 + \overline{\mathbf{z}}^{(\ell-1)m}) \\ &= \mathbf{x}^0 + \overline{\mathbf{z}}^{(\ell-1)m} + \mathbf{r}^0 - A\overline{\mathbf{z}}^{(\ell-1)m}.\end{aligned}$$

Therefore,

$$\mathbf{x}_{AAR}^{(\ell-1)m+1} \in \mathcal{K}_{(\ell-1)m}(A, \mathbf{r}^0) + A\mathcal{K}_{(\ell-1)m}(A, \mathbf{r}^0) = \mathcal{K}_{(\ell-1)m+1}(A, \mathbf{r}^0).$$

In other words, each Richardson sweep expands the Krylov subspace from which the approximation is computed, leading eventually to

$$\mathbf{x}_{AAR}^{\ell m} = \mathbf{x}^0 + \mathbf{z}^{\ell m}, \quad \mathbf{z}^{\ell m} \in \mathcal{K}_{\ell m}(A, \mathbf{r}^0).$$

The next step requires an Anderson mixing so that

$$\overline{\mathbf{x}}_{AAR}^{\ell m} = \mathbf{x}^0 - \sum_{i=1}^{\ell m} \hat{g}_i^{\ell m}(\mathbf{x}_{AAR}^i - \mathbf{x}^0),$$

where

$$\hat{\mathbf{g}}^{\ell m} = (\hat{g}_1^{\ell m}, \ldots, \hat{g}_{\ell m}^{\ell m})^T = \underset{\mathbf{g}\in\mathbb{R}^{\ell m}}{\operatorname{argmin}} \left\| \mathbf{b} - A\left[\mathbf{x}^0 - \sum_{i=1}^{\ell m} g_i(\mathbf{x}_{AAR}^i - \mathbf{x}^0)\right] \right\|_2.$$

Since $\ell m \leq \nu(A, \mathbf{r}^0)$, this still guarantees that matrices $\hat{X}_k$ and $\hat{R}_k$ have full column rank and

$$\mathcal{R}(\hat{X}_k) = \mathcal{K}_k(A, \mathbf{r}^0), \quad k \leq \ell m.$$

Since the least-squares problem minimizes the residual on the Krylov subspace $\mathcal{K}_{\ell m}(A, \mathbf{r}^0)$, we have that

$$\overline{\mathbf{x}}_{AAR}^{\ell m} = \mathbf{x}_G^{\ell m}.$$

Now we analyze what would happen if the stagnation did not occur. If this were the case, then the index $\nu(A, \mathbf{r}^0)$ would be either associated with a simple Richardson's step or with an iteration where both a Richardson's step and an Anderson mixing are performed. Because of this, any following Richardson's step would not expand the dimension of the column space of $\hat{X}_k$ anymore, meaning that

$$\mathcal{R}(\hat{X}_k) = \mathcal{K}_{\nu(A,\mathbf{r}^0)}(A, \mathbf{r}^0), \quad k \geq \nu(A, \mathbf{r}^0). \tag{3.4}$$

Denoting by $tm \geq \nu(A, \mathbf{r}^0)$ the first iteration performing an Anderson mixing following the iteration with index equal to $\nu(A, \mathbf{r}^0)$, we have

$$\overline{\mathbf{x}}_{AAR}^{tm} = \mathbf{x}^0 - \sum_{i=1}^{tm} \hat{g}_i^{tm}(\mathbf{x}_{AAR}^i - \mathbf{x}^0),$$

where

$$\hat{\mathbf{g}}^{tm} = (\hat{g}_1^{tm}, \ldots, \hat{g}_{tm}^{tm})^T = \underset{\mathbf{g} \in \mathbb{R}^{tm}}{\operatorname{argmin}} \left\| \mathbf{b} - A\left[\mathbf{x}^0 - \sum_{i=1}^{tm} g_i(\mathbf{x}_{AAR}^i - \mathbf{x}^0)\right] \right\|_2.$$

Because of the relation in (3.4), the least-squares problem minimizes the residual on the Krylov subspace $\mathcal{K}_{\nu(A,\mathbf{r}^0)}(A, \mathbf{r}^0)$. Therefore,

$$\overline{\mathbf{x}}_{AAR}^{tm} = \mathbf{x}_G^{\nu}(A, \mathbf{r}^0) = \mathbf{x},$$

which means that both Full GMRES and Full AAR converge to the exact solution of the linear system.

Note that the statement of the theorem and the proof are valid for any non-zero value of ω and β_k's. $\square$

Remark 2. Theorem 3.2.1 guarantees periodic equivalence (the equivalence occurs only after every Anderson mixing) between Full AAR and Full GMRES if there is no stagnation in the history of approximations computed by Full GMRES. This means that in absence of stagnation, Full AAR is periodically an oblique projection method (for more details, see pp. 129 in [52]). Therefore, at the iteration $k = \ell m \leq \eta_G$ we obtain

$$\mathcal{V}_{\ell m} = \mathcal{K}_{\ell m}(A, \mathbf{r}^0) = \mathcal{R}(X_{\ell m}) = \mathcal{R}(\hat{X}_{\ell m}), \quad \dim(\mathcal{V}_{\ell m}) = p = \ell m$$

and

$$\mathcal{W}_{\ell m} = A\mathcal{K}_{\ell m}(A, \mathbf{r}^0) = A\mathcal{R}(X_{\ell m}) = A\mathcal{R}(\hat{X}_{\ell m}) = \mathcal{R}(R_{\ell m}) = \mathcal{R}(\hat{R}_{\ell m}),$$

with $\dim(\mathcal{W}_{\ell m}) = p = \ell m$.

If Full GMRES stagnates, it is known that this breaks the equivalence between Full GMRES and Full AR, since the latter would never recover from the stagnation (see [47] and [61] for more details). As concerns Full AAR, the situation is more complex. Next, we are going to show that Full AAR is more robust to stagnation than Full AR, meaning that under certain circumstances the stagnation does not prevent Full AAR from converging to the solution of (2.1).

Theorem 3.2.2. *Consider Alternating Anderson-Richardson with a periodic updating interval between two consecutive Anderson mixing steps of length m. For an integer s, denote by $\overline{\mathbf{x}}_{AAR}^{sm}$ the approximate solution to* (2.1) *obtained by Full AAR after the Anderson mixing at the* (sm)*th iteration and denote $\mathbf{x}_G^{sm}$ the solution computed via Full GMRES at the* (sm)*th iteration. One of the following three cases holds:*

1. *if* $(\ell - 1)m = \eta_G$ *and* $\mathbf{x}_G^{(\ell-1)m} \neq \mathbf{x}_G^{\ell m}$, *then*

$$\overline{\mathbf{x}}_{AAR}^{sm} = \mathbf{x}_G^{sm}, \quad \textit{when} \quad s \geq \ell \quad \textit{and} \quad sm < \nu(A, \mathbf{r}^0),$$

or

$$\overline{\mathbf{x}}^{sm}_{AAR} = \mathbf{x}_G^{\nu(A,\mathbf{r}^0)} = \mathbf{x}, \quad when \quad s \geq \ell \quad and \quad sm \geq \nu(A,\mathbf{r}^0);$$

2. if $(\ell-1)m < \eta_G < \ell m$ *and* $\mathbf{x}_G^{\ell m} \neq \mathbf{x}_G^{(\ell+1)m}$, *then*

$$\overline{\mathbf{x}}^{sm}_{AAR} = \mathbf{x}_G^{sm}, \quad when \quad s \geq \ell \quad and \quad sm < \nu(A,\mathbf{r}^0),$$

or

$$\overline{\mathbf{x}}^{sm}_{AAR} = \mathbf{x}_G^{\nu(A,\mathbf{r}^0)} = \mathbf{x}, \quad when \quad s \geq \ell \quad and \quad sm \geq \nu(A,\mathbf{r}^0);$$

3. if $\eta_G = (\ell-1)m$ *and* $\mathbf{x}_G^{(\ell-1)m} = \mathbf{x}_G^{\ell m}$, *then*

$$\overline{\mathbf{x}}^{sm}_{AAR} = \mathbf{x}_G^{(\ell-1)m}, \quad when \quad s \geq (\ell-1) \quad .$$

Proof. The three items in the statement of the theorem deal with three possible scenarios of a stagnation in Full GMRES. Theorem 3.2.1 has already shown that in the absence of stagnation Full AAR and Full GMRES provide the same solution at the end of each periodic interval. The equivalence between Full GMRES and Full AAR holds after the stagnation only if the number of consecutive Richardson sweeps in Full AAR exceeds the extension of stagnation in Full GMRES. If Full GMRES stagnates for a number of iterations at least equal to the number of consecutive Richardson sweeps in Full AAR, then the equivalence between the algorithms is lost. The main three possible scenarios are represented in Figure 3.1 and they coincide with the three cases listed in the statement of the theorem. We are now going to address each of these cases separately.

1. Let us assume at first that no stagnation occurs before the iteration with index $k = (\ell-1)m$. By Theorem 3.2.1 we know that every time an Anderson

acceleration is computed by Full AAR the following holds:

$$\overline{\mathbf{x}}_{AAR}^{sm} = \mathbf{x}_G^{sm}, \quad sm \leq \eta_G.$$

If Full GMRES stagnates between the iterations with index $k = (\ell - 1)m$ and the successive one, then

$$\overline{\mathbf{x}}_{AAR}^{(\ell-1)m} = \mathbf{x}_G^{(\ell-1)m}$$

and

$$\mathbf{x}_G^{(\ell-1)m} = \mathbf{x}_G^{(\ell-1)m+1}.$$

Let us assume that the stagnation of Full GMRES lasts only for the successive q iterations where $q < m$. Therefore

$$\mathbf{x}_G^{(\ell-1)m} = \mathbf{x}_G^{(\ell-1)m+1} = \cdots = \mathbf{x}_G^{(\ell-1)m+q} \neq \cdots \neq \mathbf{x}_G^{\ell m}.$$

Full AAR performs standard Richardson sweeps at the iterations indexed by

$$k = (\ell - 1)m + 1, \ldots, \ell m.$$

In fact, the approximation $\mathbf{x}_{AAR}^{(\ell-1)m+1}$ is

$$\begin{aligned}
\mathbf{x}_{AAR}^{(\ell-1)m+1} &= \overline{\mathbf{x}}_{AAR}^{(\ell-1)m} + \mathbf{b} - A\overline{\mathbf{x}}_{AAR}^{(\ell-1)m} \\
&= \mathbf{x}^0 + \overline{\mathbf{z}}^{(\ell-1)m} + \mathbf{b} - A(\mathbf{x}^0 + \overline{\mathbf{z}}^{(\ell-1)m}) \\
&= \mathbf{x}^0 + \overline{\mathbf{z}}^{(\ell-1)m} + \mathbf{r}^0 - A\overline{\mathbf{z}}^{(\ell-1)m},
\end{aligned}$$

implying that

$$\mathbf{x}_{AAR}^{(\ell-1)m+1} \in \mathcal{K}_{(\ell-1)m}(A, \mathbf{r}^0) + A\mathcal{K}_{(\ell-1)m}(A, \mathbf{r}^0) = \mathcal{K}_{(\ell-1)m+1}(A, \mathbf{r}^0).$$

In other words, each Richardson sweep expands the Krylov subspace from which the approximation is computed, leading eventually to

$$\mathbf{x}_{AAR}^{\ell m} = \mathbf{x}^0 + \mathbf{z}^{\ell m}, \quad \mathbf{z}^{\ell m} \in \mathcal{K}_{\ell m}(A, \mathbf{r}^0).$$

The next step requires an Anderson mixing so that

$$\overline{\mathbf{x}}_{AAR}^{\ell m} = \mathbf{x}^0 - \sum_{i=1}^{\ell m} \hat{g}_i^{\ell m}(\mathbf{x}_{AAR}^i - \mathbf{x}^0),$$

where

$$\hat{\mathbf{g}}^{\ell m} = (\hat{g}_1^{\ell m}, \dots, \hat{g}_{\ell m}^{\ell m})^T = \underset{\mathbf{g} \in \mathbb{R}^{\ell m}}{\operatorname{argmin}} \left\| \mathbf{b} - A\left[\mathbf{x}^0 - \sum_{i=1}^{\ell m} g_i(\mathbf{x}_{AAR}^i - \mathbf{x}^0)\right] \right\|_2.$$

If $\ell m < \nu(A, \mathbf{r}^0)$, then $\hat{X}_{\ell m}$ has full columns rank and in particular

$$\mathcal{R}(\hat{X}_k) = \mathcal{K}_k(A, \mathbf{r}^0), \quad k \leq \ell m.$$

Since the least-squares problem minimizes the residual on the Krylov subspace $\mathcal{K}_{\ell m}(A, \mathbf{r}^0)$, we have that

$$\overline{\mathbf{x}}_{AAR}^{\ell m} = \mathbf{x}_G^{\ell m}.$$

If $\ell m \geq \nu(A, \mathbf{r}^0)$, then

$$\mathcal{R}(\hat{X}_{\ell m}) = \mathcal{K}_{\nu(A,\mathbf{r}^0)}(A, \mathbf{r}^0).$$

This means that the least-squares problem solved at the iteration ℓm minimizes the residual on the Krylov subspace $\mathcal{K}_{\nu(A,\mathbf{r}^0)}(A, \mathbf{r}^0)$, leading to

$$\overline{\mathbf{x}}_{AAR}^{\ell m} = \mathbf{x}_G^{\nu(A,\mathbf{r}^0)} = \mathbf{x}.$$

In the situation where $\ell m < \nu(A, \mathbf{r}^0)$, the performance of Full AAR after the

(ℓm)th iteration is equivalent to the performance of Full AAR using directly $\overline{\mathbf{x}}_{AAR}^{\ell m}$ as initial guess. Therefore, from the iteration index $k = \ell m$ on, Theorem 3.2.1 can be applied again using $\overline{\mathbf{x}}_{AAR}^{\ell m}$ as initial guess, eventually proving the conclusion for Case 1. This means that the type of stagnation considered in Case 1 does not impact the performance of Full AAR.

2. Let us assume now that the stagnation happens in the history of Full GMRES at an iteration with index $k = (\ell - 1)m + q$ where $q < m$. Full AAR performs Richardson steps from iteration $k = (\ell-1)m+q$ through $k = \ell m$, leading thus to

$$\mathbf{x}_{AAR}^{\ell m} = \mathbf{x}^0 + \mathbf{z}^{\ell m},$$

where $\mathbf{z}^{\ell m} \in \mathcal{K}_{\ell m}(A, \mathbf{r}^0)$. If $\ell m < \nu(A, \mathbf{r}^0)$, then $\hat{X}_{\ell m}$ has full column rank and in particular

$$\mathcal{R}(\hat{X}_k) = \mathcal{K}_k(A, \mathbf{r}^0), \quad k \leq \ell m.$$

Since the least-squares problem at the (ℓm)th iteration minimizes the residual on the Krylov subspace $\mathcal{K}_{\ell m}(A, \mathbf{r}^0)$, we have that

$$\overline{\mathbf{x}}_{AAR}^{\ell m} = \mathbf{x}_G^{\ell m}.$$

If $\ell m \geq \nu(A, \mathbf{r}^0)$, then

$$\mathcal{R}(\hat{X}_{\ell m}) = \mathcal{K}_{\nu(A,\mathbf{r}^0)}(A, \mathbf{r}^0).$$

This means that the least-squares problem solved at the iteration ℓm minimizes the residual on the Krylov subspace $\mathcal{K}_{\nu(A,\mathbf{r}^0)}(A, \mathbf{r}^0)$, leading to

$$\overline{\mathbf{x}}_{AAR}^{\ell m} = \mathbf{x}_G^{\nu(A,\mathbf{r}^0)} = \mathbf{x}.$$

If $\ell m < \nu(A, \mathbf{r}^0)$, the situation is similar to Case 1 from the iteration ℓm on.

The only difference is that the iteration index is shifted forward by m. In fact, here we have a stagnation at the iteration ℓm and we are assuming that $\mathbf{x}_G^{(\ell+1)m} \neq \mathbf{x}_G^{\ell m}$, whereas in Case 1 we were assuming that we had a stagnation at the iteration $(\ell-1)m$ and $\mathbf{x}_G^{\ell m} \neq \mathbf{x}_G^{(\ell-1)m}$. Using the same reasoning as in Case 1 but with the indices shifted forward by m, we can prove the statement for Case 2. Therefore, Full AAR converges to the solution of (2.1) even for the type of stagnation considered in Case 2.

3. We now consider the most challenging case of stagnation for Full AAR. This occurs when the stagnation in the history of Full GMRES lasts for a number of iterations equal to $q \geq m$. For ease of exposition, we are going to adopt the definition of Anderson mixing as described in Equation (2.41). Since Full AAR involves all the previous updates in the Anderson mixing, we know that the formulations (2.37) and (2.41) are equivalent.

   If there is no stagnation up to the iteration $k = (\ell-1)m$, we have that

$$\overline{\mathbf{x}}_{AAR}^{(\ell-1)m} = \mathbf{x}_G^{(\ell-1)m}.$$

   Therefore, from the definition of Anderson mixing in Equation (2.41) it follows that

$$\overline{\mathbf{x}}_{AAR}^{(\ell-1)m} = \mathbf{x}^0 - \sum_{i=1}^{(\ell-1)m} \hat{g}_i^{(\ell-1)m}(\mathbf{x}_{AAR}^i - \mathbf{x}^0) = \mathbf{x}^0 + \overline{\mathbf{z}}^{(\ell-1)m},$$

   where the vector

$$\overline{\mathbf{z}}^{(\ell-1)m} = -\sum_{i=1}^{(\ell-1)m} \hat{g}_i^{(\ell-1)m}(\mathbf{x}_{AAR}^i - \mathbf{x}^0) \in \mathcal{K}_{(\ell-1)m}(A, \mathbf{r}^0)$$

   is computed so as to minimize the ℓ^2-norm of the residual. After computing the solution to the least-squares problem, Full AAR performs Richardson steps from iteration $k = (\ell - 1)m + 1$ through iteration $k = \ell m$. We thus obtain

$$\mathbf{x}_{AAR}^{\ell m} = \mathbf{x}^0 + \mathbf{z}^{\ell m},$$

where $\mathbf{z}^{\ell m} \in \mathcal{K}_{\ell m}(A, \mathbf{r}^0)$. Under the hypotheses that $\mathbf{x}_G^{(\ell-1)m} = \mathbf{x}_G^{\ell m}$, it follows that

$$\ell m < \nu(A, \mathbf{r}^0).$$

Therefore, the matrix $\hat{X}_{\ell m}$ has full column rank and in particular

$$\mathcal{R}(\hat{X}_k) = \mathcal{K}_k(A, \mathbf{r}^0), \quad k \leq \ell m.$$

Therefore, the least-squares problem at the iteration ℓm computes $\overline{\mathbf{z}}^{\ell m} \in \mathcal{K}_{\ell m}(A, \mathbf{r}^0)$ such that

$$\overline{\mathbf{z}}^{\ell m} = \underset{\mathbf{z} \in \mathcal{K}_{\ell m}(A, \mathbf{r}^0)}{\operatorname{argmin}} \|\mathbf{b} - A(\mathbf{x}^0 + \mathbf{z})\|_2.$$

However, we are assuming that Full GMRES is still stagnating at the iteration $k = \ell m$. Therefore,

$$\overline{\mathbf{x}}_{AAR}^{\ell m} = \mathbf{x}^0 + \overline{\mathbf{z}}^{\ell m} = \mathbf{x}_G^{\ell m} = \mathbf{x}_G^{(\ell-1)m} = \overline{\mathbf{x}}_{AAR}^{(\ell-1)m}$$

and

$$\begin{aligned}
\mathbf{x}_{AAR}^{\ell m+1} &= \overline{\mathbf{x}}_{AAR}^{\ell m} + \beta_{\ell m+1}(\mathbf{b} - A\overline{\mathbf{x}}_{AAR}^{\ell m}) \\
&= \overline{\mathbf{x}}_{AAR}^{(\ell-1)m} + \beta_{\ell m+1}(\mathbf{b} - A\overline{\mathbf{x}}_{AAR}^{(\ell-1)m}) \\
&= \overline{\mathbf{x}}_{AAR}^{(\ell-1)m} + \beta_{\ell m+1}(\mathbf{b} - A(\mathbf{x}^0 + \overline{\mathbf{z}}^{(\ell-1)m})) \\
&= \overline{\mathbf{x}}_{AAR}^{(\ell-1)m} + \beta_{\ell m+1}\mathbf{r}^0 - \beta_{\ell m+1}A\overline{\mathbf{z}}^{(\ell-1)m} \\
&= \mathbf{x}^0 + \overline{\mathbf{z}}^{(\ell-1)m} + \beta_{\ell m+1}\mathbf{r}^0 - \beta_{\ell m+1}A\overline{\mathbf{z}}^{(\ell-1)m} \\
&= \mathbf{x}^0 + \beta_{\ell m+1}\mathbf{r}^0 + (I - \beta_{\ell m+1}A)\overline{\mathbf{z}}^{(\ell-1)m}.
\end{aligned} \tag{3.5}$$

This means that

$$\mathbf{x}_{AAR}^{\ell m+1} - \mathbf{x}^0 \in \mathcal{K}_{(\ell-1)m+1}(A, \mathbf{r}^0).$$

At each iteration with index

$$k = \ell m + j, \quad j = 2, \dots, m,$$

a Richardson sweep is computed so that

$$\mathbf{x}_{AAR}^{\ell m+j} = \mathbf{x}_{AAR}^{\ell m+j-1} + \omega(\mathbf{b} - A\mathbf{x}_{AAR}^{\ell m+j-1}),$$

which leads to

$$\mathbf{x}_{AAR}^{\ell m+j} - \mathbf{x}^0 \in \mathcal{K}_{(\ell-1)m+j}, \quad 1 \le j \le m.$$

Given the subspace

$$\mathcal{L}_k = \operatorname{span}\{(\mathbf{x}_{AAR}^1 - \mathbf{x}^0), (\mathbf{x}_{AAR}^2 - \mathbf{x}^0), \dots, (\mathbf{x}_{AAR}^k - \mathbf{x}^0)\}$$

we have that

$$\mathcal{L}_{\ell m+j} = \mathcal{K}_{\ell m}(A, \mathbf{r}^0), \qquad 1 \le j \le m.$$

Because of this, the Anderson mixing at $k = (\ell+1)m$ computes an approximation $\overline{\mathbf{x}}_{AAR}^{(\ell+1)m}$ that minimizes the residual onto $\mathcal{L}_{(\ell+1)m} = \mathcal{K}_{\ell m}(A, \mathbf{r}^0)$. Therefore,

$$\overline{\mathbf{x}}_{AAR}^{(\ell+1)m} = \overline{\mathbf{x}}_{AAR}^{\ell m} = \overline{\mathbf{x}}_{AAR}^{(\ell-1)m}.$$

By repeating the process for a generic periodic interval like the one described by

$$k = sm + 1, \dots, (s+1)m,$$

we obtain

$$\mathbf{x}_{AAR}^{sm+j} - \mathbf{x}^0 \in \mathcal{K}_{(\ell-1)m+j}(A, \mathbf{r}^0), \quad s \geq \ell - 1, \quad 1 \leq j \leq m.$$

Therefore, Full AAR stagnates from the iteration $k = \ell m$ on, without ever recovering because

$$\mathcal{L}_k = \mathcal{L}_{\ell m} \neq \mathcal{K}_{\ell m+1}(A, \mathbf{r}^0), \quad k \geq \ell m.$$

Hence,

$$\overline{\mathbf{x}}_{AAR}^{sm} = \overline{\mathbf{x}}_{AAR}^{(\ell-1)m} = \mathbf{x}_G^{(\ell-1)m}, \quad s \geq \ell - 1,$$

whereas Full GMRES keeps on increasing the Krylov subspace $\mathcal{K}_k(A, \mathbf{r}^0)$ used to compute $\mathbf{x}_G^k$ and it eventually converges to the solution of (2.1).

□

Remark 3. Theorem 3.2.2 shows that the convergence of Full AAR is not affected by a stagnation that occurs in the equivalent Full GMRES as long as the stagnation lasts less than m iterations. In terms of robustness against stagnation, this represents an improvement over Full AR. In fact, a theoretical discussion in [47] and [61] shows that Full AR cannot recover from any stagnation. This happens because Full AR solves a least-squares problem to minimize the residual at each iteration, whereas Full AAR solves a least-squares problem only at periodic intervals of length m. This suggests that there are two advantages in solving least-squares problem less frequently through the insertion of multiple relaxations steps in between. On the one hand, it alleviates the computational burden because there is no need to solve a least-squares problem at each iteration but only after a batch of m Richardson sweeps, which reduces computational complexity and increases locality. On the other hand, it also enhances the robustness against stagnation by making their occurrence less likely.

This property of Full AAR is likely to be inherited by truncated or restarted versions of Alternating Anderson-Richardson as well.

Although Full AAR and Full GMRES are mathematically equivalent at the end of each periodic interval of length m, the bases constructed by the two algorithms for the projection steps are not the same. Only the first vector is the same up to a scaling factor. In fact, the first vector of the basis built by Full GMRES is

$$\mathbf{v}_1 = \frac{\mathbf{r}^0}{\|\mathbf{r}^0\|_2}$$

and the first vector of the basis built by Full AAR is

$$\mathbf{x}^1_{AAR} - \mathbf{x}^0 = \omega \mathbf{r}^0.$$

We aim now to compare the second vector constructed by each algorithm. We recall that the non-normalized second vector of the Full GMRES basis is

$$\mathbf{w}_1 = \left[I - \frac{\mathbf{r}^0}{\|\mathbf{r}^0\|_2} \left(\frac{\mathbf{r}^0}{\|\mathbf{r}^0\|_2} \right)^T \right] \frac{A\mathbf{r}^0}{\|\mathbf{r}^0\|_2}.$$

As concerns Full AAR, we need to carry out two different computations, depending on which implementation of Anderson mixing is adopted. The approximation computed by Full AAR at the second iteration is

$$\begin{aligned}
\mathbf{x}^2_{AAR} &= \mathbf{x}^1_{AAR} + \omega(\mathbf{b} - A\mathbf{x}^1) \\
&= \mathbf{x}^0 + \omega \mathbf{r}^0 + \omega(\mathbf{b} - A(\mathbf{x}^0 + \omega \mathbf{r}^0)) \\
&= \mathbf{x}^0 + 2\omega \mathbf{r}^0 - \omega^2 A \mathbf{r}^0.
\end{aligned}$$

If the Anderson mixing were defined as in Equation (2.37), then the second vector of

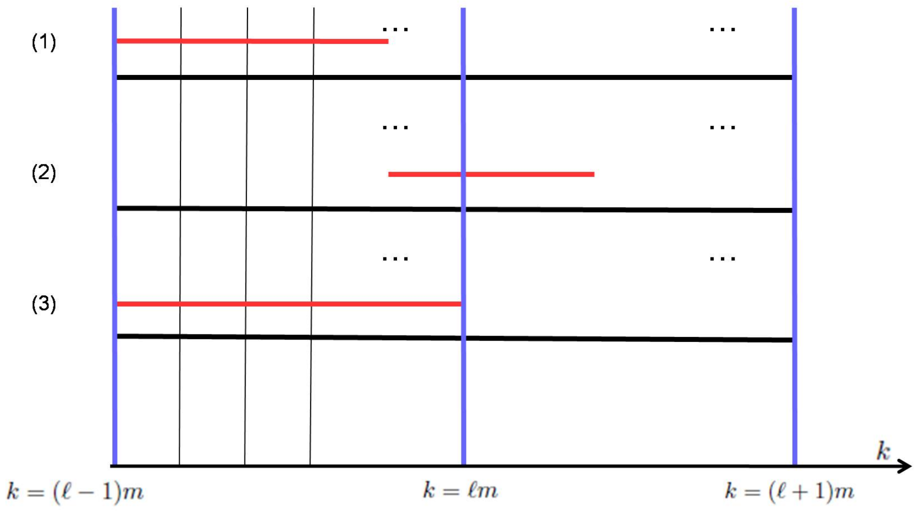

Figure 3.1: Representation of the three main cases of stagnation: case (1) at the top, case (2) at the center and case (3) at the bottom. The x-axis displays the iteration index k. The use of different thickness and colors for the vertical lines allows us to differentiate the iteration based on how Full AAR updates the approximation. The continuous thick vertical blue lines at the extremes of the figure and at the center indicate the points where three consecutive Anderson mixing steps are performed (at $k = (\ell - 1)m$, $k = \ell m$ and $k = (\ell + 1)m$). The fine vertical black lines represent iterations where Full AAR performs a standard Richardson sweep. As concerns Full GMRES, it performs the same algebraic operations across any iterations. The continuous thick horizontal red segments represent the stagnation of Full GMRES and their length coincides with the duration of the stagnation. The red lines stop where Full GMRES recovers from the stagnation. Case (1) deals with the situation where the stagnation starts at the iteration $k = (\ell - 1)m$ and stops before the iteration $k = \ell m$. Case (2) deals with the situation where the stagnation starts at an iteration index between $k = (\ell - 1)m$ and $k = \ell m$, extending up to another iteration index between $k = \ell m$ and $k = (\ell + 1)m$. The third case represents the situation where the stagnation starts at the iteration $k = (\ell - 1)m$ and it lasts till the iteration $k = \ell m$ is reached.

the basis built by Full AAR would be

$$\mathbf{x}_{AAR}^{2} - \mathbf{x}_{AAR}^{1} = \omega(I - \omega A)\mathbf{r}^{0}.$$

If the Anderson mixing were defined as in Equation (2.41) instead, this would lead to an expansion of the basis through the vector

$$\mathbf{x}_{AAR}^{2} - \mathbf{x}^{0} = \omega(2I - \omega A)\mathbf{r}^{0}.$$

It is possible to see that $\mathbf{w}_1$, $\mathbf{x}_{AAR}^{2} - \mathbf{x}_{AAR}^{1}$ and $\mathbf{x}_{AAR}^{2} - \mathbf{x}^{0}$ identify different directions. In general this difference propagates also to the other vectors used to expand the basis leading to

$$\begin{aligned}
&\operatorname{span}\{\mathbf{v}_k\} \neq \operatorname{span}\{\mathbf{x}_{AAR}^{k} - \mathbf{x}_{AAR}^{k-1}\},\\
&\operatorname{span}\{\mathbf{v}_k\} \neq \operatorname{span}\{\mathbf{x}_{AAR}^{k} - \mathbf{x}^{0}\},\\
&\operatorname{span}\{\mathbf{x}_{AAR}^{k} - \mathbf{x}_{AAR}^{k-1}\} \neq \operatorname{span}\{\mathbf{x}_{AAR}^{k} - \mathbf{x}^{0}\}, \quad k \geq 2.
\end{aligned}$$

Therefore, in general the following holds

$$\begin{aligned}
&\operatorname{span}\{\mathbf{v}_\ell, \dots, \mathbf{v}_k\} \neq \operatorname{span}\{\mathbf{x}_{AAR}^{\ell} - \mathbf{x}_{AAR}^{\ell-1}, \dots, \mathbf{x}_{AAR}^{k} - \mathbf{x}_{AAR}^{k-1}\},\\
&\operatorname{span}\{\mathbf{v}_\ell, \dots, \mathbf{v}_k\} \neq \operatorname{span}\{\mathbf{x}_{AAR}^{\ell} - \mathbf{x}^{0}, \dots, \mathbf{x}_{AAR}^{k} - \mathbf{x}^{0}\},\\
&\operatorname{span}\{\mathbf{x}_{AAR}^{\ell} - \mathbf{x}_{AAR}^{\ell-1}, \dots, \mathbf{x}_{AAR}^{k} - \mathbf{x}_{AAR}^{k-1}\} \neq \operatorname{span}\{\mathbf{x}_{AAR}^{\ell} - \mathbf{x}^{0}, \dots, \mathbf{x}_{AAR}^{k} - \mathbf{x}^{0}\},
\end{aligned} \tag{3.6}$$

with $1 < \ell \leq k$.

From a practical viewpoint, a fixed value of p across the iterations is recommended, in order to alleviate the computational cost of each least-squares problem. In case of a truncation of the basis, Alternating Anderson-Richardson leads to a variant that resembles Truncated GMRES. We refer to this variant as Truncated AAR. However, the inequalities in (3.6) show that Truncated AAR and Truncated GMRES are not equivalent in exact arithmetic, since the subspaces adopted for the projection differ.

We restrict now the attention to what happens inside a periodic interval that separates two consecutive Anderson accelerations. The goal is to explore the geometric properties of the subspace used by AAR. For the rest of this section we discuss only the variant of AAR that uses the Anderson mixing as in Equation (2.37). In particular, we will make considerations about the ranges of matrices X_k and R_k defined in Formulas (2.38) and (2.39), respectively.

Assuming that $k = \ell$ is the index associated with an Anderson mixing, we have

$$\begin{aligned}
\mathbf{r}^{\ell+1}_{AAR} &= (I - \beta_\ell A)\bar{\mathbf{r}}^{\ell}_{AAR} \\
&= (I - \beta_\ell A)(I - \Pi_\ell)\mathbf{r}^{\ell}_{AAR} \\
&= \mathbf{r}^{\ell}_{AAR} - \beta_\ell A\mathbf{r}^{\ell}_{AAR} - \Pi_\ell \mathbf{r}^{\ell}_{AAR} + \beta_\ell A \Pi_\ell \mathbf{r}^{\ell}_{AAR}.
\end{aligned}$$

The vector $\mathbf{r}^{\ell+1}_{AAR}$ is the residual associated with the first Richardson sweep that follows the Anderson mixing. Therefore,

$$\begin{aligned}
\mathbf{r}^{\ell+1}_{AAR} - \mathbf{r}^{\ell}_{AAR} &= -\beta_\ell A\mathbf{r}^{\ell}_{AAR} - \Pi_\ell \mathbf{r}^{\ell}_{AAR} + \beta_\ell A \Pi_\ell \mathbf{r}^{\ell}_{AAR} \\
&= -[\Pi_\ell \mathbf{r}^{\ell}_{AAR} + \beta_\ell A\mathbf{r}^{\ell}_{AAR} - \beta_\ell A \Pi_\ell \mathbf{r}^{\ell}_{AAR}] \\
&= -[\Pi_\ell + \beta_\ell A(I - \Pi_\ell)]\mathbf{r}^{\ell}_{AAR}.
\end{aligned}$$

We recall the relation that holds for residual vectors of consecutive relaxation sweeps:

$$\mathbf{r}^{\ell}_{AAR} = (I - \omega A)\mathbf{r}^{\ell-1}_{AAR} = \cdots = (I - \omega A)^m \mathbf{r}^{\ell-m}_{AAR}, \tag{3.7}$$

and

$$\begin{aligned}\mathbf{r}_{AAR}^{\ell} - \mathbf{r}_{AAR}^{\ell-1} &= -\omega A \mathbf{r}_{AAR}^{\ell-1} \\ &= -\omega A(I - \omega A)\mathbf{r}_{AAR}^{\ell-2} \\ &= \cdots \\ &= -\omega A(I - \omega A)^{m-1}\mathbf{r}_{AAR}^{\ell-m}.\end{aligned}$$

This implies that the matrix $R_{\ell+1}$ becomes

$$R_{\ell+1} = \big[-\omega A(I-\omega A)^{m-p+1}\mathbf{r}_{AAR}^{\ell-m}, \ldots, -\omega A(I-\omega A)^{m-1}\mathbf{r}_{AAR}^{\ell-m}, \mathbf{r}_{AAR}^{\ell+1} - \mathbf{r}_{AAR}^{\ell}\big],$$

with

$$\mathbf{r}_{AAR}^{\ell+1} - \mathbf{r}_{AAR}^{\ell} = -[\Pi_\ell - \beta_\ell A(I - \Pi_\ell)](I_n - \omega A)^m \mathbf{r}_{AAR}^{\ell-m}.$$

Note that we use MATLAB notation for columns. In particular, the explicit representation of $\mathbf{r}_{AAR}^{\ell+1} - \mathbf{r}_{AAR}^{\ell}$ in terms of $\mathbf{r}_{AAR}^{\ell-m}$ highlights two distinct components:

$$\Pi_\ell (I - \omega A)^m \mathbf{r}_{AAR}^{\ell-m} \in \mathcal{R}(\Pi_\ell),$$

and

$$\beta_\ell A(I - \Pi_\ell)(I - \omega A)^m \mathbf{r}_{AAR}^{\ell-m}.$$

The second component consists of a vector lying in $[\mathcal{R}(\Pi_\ell)]^\perp$, which is then multiplied by A. If the dominant eigenvalue (eigenvalue with maximum modulus) of A is semi-simple and m sufficiently large, then the subspace $[\mathcal{R}(\Pi_k)]^\perp$ tends to be orthogonal to the dominant eigenvector. This highlights a resemblance between AAR and Deflated GMRES with restart (GMRES-DR for short) [14,44]. GMRES-DR aims to enrich the standard Krylov subspace with approximations of the eigenvectors related to small eigenvalues of A. In fact, the performance of GMRES is well known to be adversely affected by possible eigenvalues of A close to zero and an augmentation of the Krylov basis via eigenvectors (or their approximations) associated with small eigenvalues has shown significant improvements [44]. In the case of AAR, it seems this phenomenon is already automatically occurring in an approximate form, by projecting the residual onto the subspace $[\mathcal{R}(\Pi_\ell)]^\perp$ and using it to update the matrix $R_{\ell+1}$. Therefore, it is strongly recommended that p be taken greater than m, so that the window of previous approximations used by the Anderson mixing is wide enough to retain some information about $[\mathcal{R}(\Pi_\ell)]^\perp$.

Algorithm 7: GMRES-DR

```
Data: $\mathbf{x}^0$, $p$, $m$, $tol$
Result: $\mathbf{x}^{(\ell-1)m}$
1  $\ell = 1$;
2  Compute $\mathbf{r}^0 = \mathbf{b} - A\mathbf{x}^0$;
3  Compute $\beta = \|\mathbf{r}^0\|_2$;
4  Compute $\mathbf{v}_1 = \frac{\mathbf{r}^0}{\beta}$;
5  while $\frac{\|\mathbf{r}^{(\ell-1)m}\|_2}{\|\mathbf{b}\|_2} > tol$ do
6      for $k = 1, \dots, m$ do
7          Set $\mathbf{w}_k = \begin{cases} \mathbf{v}_k & \text{if } k \leq m-p \\ \mathbf{u}_{k-m+p} \text{ (eigenvector)} & \text{otherwise} \end{cases}$;
8          Compute $\mathbf{w}_k = A\mathbf{v}_k$;
9          for $i = 1, \dots, k$ do
10             $h_{ik} = \langle \mathbf{w}_k, \mathbf{v}_i \rangle$;
11             $\mathbf{w}_k = \mathbf{w}_k - h_{ik}\mathbf{v}_i$;
12         end
13         $h_{k+1,k} = \|\mathbf{w}_k\|_2$. If $h_{k+1,k} = 0$ set $m = k$ and go to 17;
14         $\mathbf{v}_{k+1} = \frac{\mathbf{w}_k}{h_{k+1,k}}$;
15     end
16     Define the $(m+1) \times m$ Hessenberg matrix $\overline{H}_{\ell m} = \{h_{ij}\}_{1 \leq i \leq m, 1 \leq j \leq m}$;
17     Compute $\mathbf{y}^{\ell m}$, the minimizer of $\|\beta \mathbf{e}_1 - \overline{H}_{\ell m}\mathbf{y}\|_2$, and $\mathbf{x}^{\ell m} = \mathbf{x}^0 + V_{\ell m}\mathbf{y}^{\ell m}$ ;
18     Set $\mathbf{x}^0 = \mathbf{x}^{\ell m}$;
19     Compute $\mathbf{r}^{\ell m} = \mathbf{b} - A\mathbf{x}^{\ell m}$;
20     $\beta = \|\mathbf{r}^{\ell m}\|_2$;
21     Set $\mathbf{v}_1 = \frac{\mathbf{r}^{\ell m}}{\beta}$ ;
22     $\ell = \ell + 1$;
23 end
```

Besides deflation, several other approaches have been proposed in the literature to improve the performance of Restarted GMRES. In [6] the authors attempt to accelerate Restarted GMRES by enriching the Krylov subspace through an ad-hoc set of linearly independent vectors, given by the difference of solution vectors computed at the end of two consecutive restarting cycles. This strategy aims at accelerating the convergence by increasing the angle among residual vectors at the end of every other cycle. The performance of these variants of GMRES is monitored in [6] through the angles $\angle(\mathbf{r}^{k+m}, \mathbf{r}^k)$ and $\angle(\mathbf{r}^{k+2m}, \mathbf{r}^k)$, where m denotes the number of iterations inside a cycle of Restarted GMRES. In fact, the angle $\angle(\mathbf{r}^{k+2m}, \mathbf{r}^k)$ may be close to zero regardless of the value attained by $\angle(\mathbf{r}^{k+m}, \mathbf{r}^k)$. In particular, situations where $\angle(\mathbf{r}^{k+2m}, \mathbf{r}^k)$ is small typically exhibit a slow convergence of Restarted GMRES. Therefore, the authors in [6] suggest that $\angle(\mathbf{r}^{k+m}, \mathbf{r}^k)$ combined with $\angle(\mathbf{r}^{k+2m}, \mathbf{r}^k)$ is a better measure to monitor the performance. We are going to follow the same principle to study the performance of Truncated AAR, stressing one more reason why $p > m$ is important to facilitate the convergence. We use again the index $k = \ell$ to refer to an iteration where the Anderson mixing is performed. For the sake of simplicity, let us temporarily assume that

$$m \leq p \leq 2m - 1.$$

The construction of the matrix $R_{\ell+m}$ thus leads to

$$\begin{aligned}
&R_{\ell+m}(:, 1 : p - m) = [\mathbf{r}_{AAR}^{\ell-(p-m)+1} - \mathbf{r}_{AAR}^{\ell-(p-m)}, \ldots, \mathbf{r}_{AAR}^{\ell} - \mathbf{r}_{AAR}^{\ell-1}],\\
&R_{\ell+m}(:, p - m + 1) = \mathbf{r}_{AAR}^{\ell+1} - \mathbf{r}_{AAR}^{\ell},\\
&R_{\ell+m}(:, p - m + 2 : p) = [\mathbf{r}_{AAR}^{\ell+2} - \mathbf{r}_{AAR}^{\ell+1}, \ldots, \mathbf{r}_{AAR}^{\ell+m} - \mathbf{r}_{AAR}^{\ell+m-1}],
\end{aligned}$$

where we use Matlab notation for blocks of columns. We study $\mathcal{R}(R_{\ell+m}(:, 1 : p-m))$ at first. By exploiting the same chain of equalities for residual vectors of successive Richardson steps as in (3.7) we obtain

$$\mathbf{r}_{AAR}^{\ell-(p-m)+1} - \mathbf{r}_{AAR}^{\ell-(p-m)} = -\omega A(I - \omega A)^{2m-p-1}\mathbf{r}_{AAR}^{\ell-m+1},$$

$$\mathbf{r}_{AAR}^{\ell-(p-m)+2} - \mathbf{r}_{AAR}^{\ell-(p-m)+1} = -\omega A(I - \omega A)^{2m-p}\mathbf{r}_{AAR}^{\ell-m+1},$$

$$\ldots$$

$$\mathbf{r}^{\ell}_{AAR} - \mathbf{r}^{\ell-1}_{AAR} = -\omega A(I-\omega A)^{m-2}\mathbf{r}^{\ell-m+1}_{AAR}.$$

Therefore,

$$\mathcal{R}(R_{\ell+m}(:,1:p-m)) = \text{span}\{A(I-\omega A)^{2m-p-1}\mathbf{r}^{\ell-m+1}_{AAR}, \ldots, A(I-\omega A)^{m-2}\mathbf{r}^{\ell-m+1}_{AAR}\}.$$

The chain of equalities in (3.7) can also be used to study $\mathcal{R}(R_{\ell+m}(:,p-m+2:p))$. In fact, the following set of equalities:

$$\mathbf{r}^{\ell+2}_{AAR} - \mathbf{r}^{\ell+1}_{AAR} = -\omega A\mathbf{r}^{\ell+1}_{AAR},$$

$$\mathbf{r}^{\ell+3}_{AAR} - \mathbf{r}^{\ell+2}_{AAR} = -\omega A(I-\omega A)\mathbf{r}^{\ell+1}_{AAR},$$

$$\cdots$$

$$\mathbf{r}^{\ell+m}_{AAR} - \mathbf{r}^{\ell+m-1}_{AAR} = -\omega A(I-\omega A)^{m-2}\mathbf{r}^{\ell+1}_{AAR}$$

leads to

$$\mathcal{R}(R_{\ell+m}(:,p-m+2:p)) = A\mathcal{K}_{m-1}(A,\mathbf{r}^{\ell+1}).$$

In conclusion, we can say that

$$\begin{aligned}\mathcal{R}(R_{\ell+m}) &\supseteq \mathcal{R}(R_{\ell+m}(:,1:p-m)) + \mathcal{R}(R_{\ell+m}(:,p-m+2:p)) \\ &= \mathcal{R}(R_{\ell+m}(:,1:p-m)) + A\mathcal{K}_{m-1}(A,\mathbf{r}^{\ell+1}).\end{aligned} \tag{3.8}$$

The relation (3.8) shows that $\mathcal{R}(R_{\ell+m})$ retains some information about $\mathbf{r}^{\ell-m+1}_{AAR}$ via

$$\mathcal{R}(R_{\ell+m}(:,1:p-m)).$$

This facilitates the linear independence between $\mathbf{r}^{\ell-m+1}_{AAR}$ and $\mathbf{r}^{\ell+m+1}_{AAR}$. Therefore, convergence is favored by preventing collinearity between residuals that are separated by two cycles, which explains the importance of having $p > m$.

In a similar way to what we already did for AR, it is possible to stress an analogy between aggregation/disaggregation methods and Alternating Anderson-Richardson. In fact, the authors in [15] mentioned the possibility for the projection step to turn a non-convergent Richardson sweep into a convergent one. Furthermore, if the subspace adopted for the projection is effective, the acceleration may even benefit multiple successive Richardson steps. Therefore, simple relaxation steps that do not define a convergent splitting may be transformed into contractions on the current residual. Let us denote with $k = \ell$ the iteration index associated with an Anderson mixing. We obtain

$$\mathbf{r}^{\ell+1+j} = \Big[\underbrace{(I - \omega A)^j (I - \beta_\ell A)}_{\text{Richardson sweeps}} \cdot \underbrace{(I - \Pi_\ell)}_{\text{Anderson acceleration}}\Big]\mathbf{r}^\ell, \quad j \leq m - 1.$$

Then the possibility mentioned by the authors in [15] for aggregation/disaggregation accelerations can also be stated with respect to AAR. In fact, it is possible that $(I - \omega A)^j (I - \beta_\ell A)(I - \Pi_\ell)$ may operate on the residual $\mathbf{r}^\ell$ in a contractive way, although the matrix $(I - \omega A)^j (I - \beta_\ell A)$ itself may not even generate a convergent splitting.

Remark 4. If R_ℓ is not a full-rank matrix, then $\mathcal{R}(R_\ell)$ is A-invariant. However, using R_ℓ to compute an Anderson mixing as in Equation (2.37) is still legitimate by solving least-squares problems. Indeed, albeit some numerical examples presented in Section 3.3 generated a matrix R_ℓ with linearly dependent columns at some iterations, this did not hinder convergence.

3.2.2 Convergence analysis for Alternating Anderson-Richardson

Next, we prove a convergence result for AAR analogous to the one found in [60, Thm.2.1] for AR.

Theorem 3.2.3. *Consider a nonsingular linear system as in Equation* (2.1)*. If the iteration matrix $H = I - A$ is such that $\|H\|_2 = c < 1$, then Alternating Anderson-Richardson converges to the solution of* (2.1) *in exact arithmetic for any choice of p and m, regardless of the initial guess. The residual norm converges to zero q-linearly and the q-factor is c.*

Proof. Denoting by $\mathbf{x}^k_{AAR}$ the Richardson approximation before the Anderson acceleration, the associated residual is

$$\mathbf{r}^k_{AAR} = H\mathbf{r}^{k-1}_{AAR},$$

hence

$$\|\mathbf{r}^k_{AAR}\|_2 \leq \|H\|_2\|\mathbf{r}^{k-1}_{AAR}\|_2 < c\|\mathbf{r}^{k-1}_{AAR}\|_2.$$

The Anderson mixing is computed to minimize the residual. By recalling the definition of Anderson mixing as in Equation (2.37) or (2.41) to compute $\overline{\mathbf{x}}^k_{AAR}$ we have $\overline{\mathbf{r}}^k_{AAR} = \mathbf{b} - A\overline{\mathbf{x}}^k_{AAR}$ and it holds that

$$\|\overline{\mathbf{r}}^k_{AAR}\|_2 \leq \|\mathbf{r}^k_{AAR}\|_2 < c\|\mathbf{r}^{k-1}_{AAR}\|_2,$$

regardless of the choice of p. This proves that the residual ℓ^2-norm converges to zero q-linearly and the q-factor is c. $\square$

Similarly to what already discussed in [60] for AR, requiring $\|H\|_2 < 1$ for AAR to converge is too restrictive and impractical in many cases. If the condition $\|H\|_2 < 1$ does not hold, Truncated AAR cannot be guaranteed to converge in general. This motivates the possible employment of a variant of the algorithm which we call *Augmented Alternating Anderson-Richardson.*

3.2.3 Augmented Alternating Anderson-Richardson

In this section we discuss a way to expand the subspace of projection used by AAR so that convergence results can be obtained for specific classes of matrices. The basic idea behind the augmentation of the subspace that we are going to present is to split a vector $\mathbf{z} \in \mathbb{R}^n$ into two components

$$\mathbf{z} = \mathbf{z}^1 + \mathbf{z}^2, \quad \mathbf{z}^1, \mathbf{z}^2 \in \mathbb{R}^n$$

so that

$$\text{span}\{\mathbf{z}^1, \mathbf{z}^2\} \supseteq \text{span}\{\mathbf{z}\}.$$

The criterion adopted to decompose the vector of interest is guided by the search of an expanded subspace where convergence of Truncated AAR can be guaranteed under certain hypotheses. In this section we consider the algorithm of Truncated AAR with $p > m$ and we focus on the implementation of Anderson mixing as in Equation (2.37). We denote with $k = \ell$ the iteration index where a Richardson sweep is performed to compute $\mathbf{x}^{\ell}_{AAR}$ followed by an Anderson mixing to compute $\overline{\mathbf{x}}^{\ell}_{AAR}$. From Equation (2.37) we obtain

$$\overline{\mathbf{x}}^{\ell}_{AAR} = \mathbf{x}^{\ell}_{AAR} - X_{\ell}\mathbf{g}^{\ell}.$$

Then a successive Richardson step computes

$$\begin{aligned}
\mathbf{x}^{\ell+1}_{AAR} &= \overline{\mathbf{x}}^{\ell}_{AAR} + \beta_{\ell}(\mathbf{b} - A\overline{\mathbf{x}}^{\ell}_{AAR}) \\
&= \mathbf{x}^{\ell}_{AAR} - X_{\ell}\mathbf{g}^{\ell} + \beta_{\ell}(\mathbf{b} - A\overline{\mathbf{x}}^{\ell}_{AAR}) \\
&= \mathbf{x}^{\ell}_{AAR} - X_{\ell}\mathbf{g}^{\ell} + \beta_{\ell}\overline{\mathbf{r}}^{\ell}_{AAR},
\end{aligned}$$

where $\overline{\mathbf{r}}^{\ell}_{AAR} = \mathbf{b} - A\overline{\mathbf{x}}^{\ell}_{AAR}$. Recalling the definition of $X_{\ell+1} \in \mathbb{R}^{n\times p}$ in (2.38), the vector used in Truncated AAR to update $X_{\ell+1}$ is $\mathbf{x}^{\ell+1}_{AAR} - \mathbf{x}^{\ell}_{AAR}$ so that

$$X_{\ell+1} = [X_\ell(:, 2:p), \mathbf{x}^{\ell+1}_{AAR} - \mathbf{x}^{\ell}_{AAR}] = [X_\ell(:, 2:p), -X_\ell\mathbf{g}^\ell + \beta_\ell\overline{\mathbf{r}}^{\ell}_{AAR}].$$

A desirable property to produce convergence results would be that the direction identified by the minimal residual $\overline{\mathbf{r}}^{\ell}_{AAR}$ be included in the subspace of projection. One way to obtain this is to replace the vector

$$-X_\ell\mathbf{g}^\ell + \beta_\ell\overline{\mathbf{r}}^{\ell}_{AAR}$$

in the last column of $X_{\ell+1}$ with the separate vectors $X_\ell\mathbf{g}^\ell$ and $\overline{\mathbf{r}}^{\ell}_{AAR}$. Obviously

$$\text{span}\{X_\ell\mathbf{g}^\ell, \overline{\mathbf{r}}^{\ell}_{AAR}\} \supseteq \text{span}\{-X_\ell\mathbf{g}^\ell + \beta_\ell\overline{\mathbf{r}}^{\ell}_{AAR}\},$$

which guarantees that no information is lost due to the separation of the original vector in two components. This allows us to replace $X_{\ell+1}$ with a new matrix $\tilde{X}_{\ell+1}$ defined as follows:

$$\tilde{X}_{\ell+1} = \begin{cases} [X_\ell(:, 2:p), X_\ell\mathbf{g}^\ell, \beta_\ell\overline{\mathbf{r}}^{\ell}_{AAR}] \in \mathbb{R}^{n\times(p+1)}, & \text{if} \quad g^\ell_1 \neq 0 \\ [X_\ell(:, 2:p), \beta_\ell\overline{\mathbf{r}}^{\ell}_{AAR}] \in \mathbb{R}^{n\times p}, & \text{if} \quad g^\ell_1 = 0 \end{cases}. \tag{3.9}$$

The distinction of the two cases in (3.9) is driven by numerical considerations. In fact, it is important to guarantee that $\tilde{X}_{\ell+1}$ have full column rank. Because of the way Anderson mixing is defined in Equation (2.37), we know that

$$X_\ell\mathbf{g}^\ell \in \mathcal{R}(X_\ell).$$

However,

$$X_\ell \mathbf{g}^\ell \notin \mathcal{R}(X_\ell(:, 2:p)), \quad \text{if} \quad g_1^\ell \neq 0.$$

Therefore, the inclusion of the vector $X_\ell \mathbf{g}^\ell$ as a column of $\tilde{X}_{\ell+1}$ does not preclude the full column rank property if $g_1^\ell \neq 0$. Moreover,

$$\bar{\mathbf{r}}_{AAR}^\ell \notin \mathcal{R}(X_\ell(:, 2:p)) + \text{span}\{X_\ell \mathbf{g}^\ell\},$$

which always legitimates the inclusion of the vector $\bar{\mathbf{r}}_{AAR}^\ell$ as a column of $\tilde{X}_{\ell+1}$. Notice also that from (3.9) we have

$$\mathcal{R}(\tilde{X}_{\ell+1}) \supseteq \mathcal{R}(X_{\ell+1}).$$

If $\tilde{X}_{\ell+1}$ has $p+1$ columns, one may shrink it to restore the initial dimension of the projected problem. In our numerical examples, we keep $\tilde{X}_{\ell+1}$ with $p+1$ columns if $g_1^\ell \neq 0$. This choice of expanding the number of columns from p to $p+1$ introduces an additional computational cost, due to the increase of the size of the least-squares problem to solve. Regardless of the arbitrary way to reduce from $p+1$ to p columns, the only thing that matters is that

$$\bar{\mathbf{r}}_{AAR}^\ell \in \mathcal{R}(\tilde{X}_{\ell+1}).$$

In fact, the direction identified by $\bar{\mathbf{r}}_{AAR}^\ell$ is going to play a central role in the following convergence result of Augmented AAR for positive definite matrices.

Theorem 3.2.4. *Consider a linear system as in Equation* (2.1). *If A is positive definite and the parameters of Augmented AAR satisfy $p \geq m$, then Augmented AAR converges to the solution of* (2.1) *in exact arithmetic, regardless of the initial guess.*

Proof. We denote by $k = \ell$ the iteration index where a Richardson sweep is performed to compute $\mathbf{x}^{\ell}_{AAR}$ followed by an Anderson mixing that produces $\overline{\mathbf{x}}^{\ell}_{AAR}$. We thus obtain

$$\overline{\mathbf{x}}^{\ell}_{AAR} = \mathbf{x}^{\ell}_{AAR} - X_{\ell}\mathbf{g}^{\ell}$$

and

$$\overline{\mathbf{r}}^{\ell}_{AAR} = \mathbf{b} - A\overline{\mathbf{x}}^{\ell}_{AAR}.$$

The proof mimics the reasoning presented in [52] (see p. 205) to prove the convergence of Restarted GMRES(m) on positive definite matrices. In particular, if $p \geq m$, then

$$\overline{\mathbf{r}}^{\ell}_{AAR} \in \mathcal{R}(\tilde{X}_{\ell+m}).$$

Therefore, $\mathcal{R}(\tilde{X}_{\ell+m})$ includes the direction of the initial residual vector at each cycle. This means that $\mathcal{R}(\tilde{X}_{\ell+m})$ identifies an hyperplane which contains the direction used by the Minimal Residual method (MR) (see p. 140 in [52]) to update the solution. Let

$$\mu = \frac{\lambda_{\min}(A + A^T)}{2}, \quad \sigma = \|A\|_2.$$

Note that $\mu \leq \sigma$. Since Augmented AAR minimizes the residual ℓ^2-norm on the subspace $\mathcal{R}(\tilde{X}_{\ell+m})$ at each Anderson mixing, the residual norm at the end of a cycle is reduced at least as much as the result of one step of MR. Therefore, in the worst case we obtain

$$\overline{\mathbf{r}}^{\ell+m}_{AAR} = \overline{\mathbf{r}}^{\ell}_{AAR} - \alpha_{\ell} A\overline{\mathbf{r}}^{\ell}_{AAR},$$

where

$$\alpha_{\ell} = \frac{\langle A\overline{\mathbf{r}}^{\ell}_{AAR}, \overline{\mathbf{r}}^{\ell}_{AAR}\rangle}{\langle A\overline{\mathbf{r}}^{\ell}_{AAR}, A\overline{\mathbf{r}}^{\ell}_{AAR}\rangle}.$$

Exploiting the definition of ℓ^2-norm of a vector, it follows that

$$\begin{aligned}\|\overline{\mathbf{r}}^{\ell+m}_{AAR}\|_2^2 &= \langle \overline{\mathbf{r}}^{\ell}_{AAR} - \alpha_\ell A\overline{\mathbf{r}}^{\ell}_{AAR}, \overline{\mathbf{r}}^{\ell}_{AAR} - \alpha_\ell A\overline{\mathbf{r}}^{\ell}_{AAR}\rangle \\ &= \langle \overline{\mathbf{r}}^{\ell}_{AAR} - \alpha_\ell A\overline{\mathbf{r}}^{\ell}_{AAR}, \overline{\mathbf{r}}^{\ell}_{AAR}\rangle - \alpha_\ell\langle \overline{\mathbf{r}}^{\ell}_{AAR} - \alpha_\ell A\overline{\mathbf{r}}^{\ell}_{AAR}, A\overline{\mathbf{r}}^{\ell}_{AAR}\rangle.\end{aligned}$$

By construction, it is known that

$$\overline{\mathbf{r}}^{\ell+m}_{AAR} \in [\mathcal{R}(\tilde{R}_{\ell+m})]^\perp,$$

where $\tilde{R}_{\ell+m} = -A\tilde{X}_{\ell+m}$. Therefore,

$$\langle \overline{\mathbf{r}}^{\ell+m}_{AAR}, A\overline{\mathbf{r}}^{\ell}_{AAR}\rangle = \langle \overline{\mathbf{r}}^{\ell}_{AAR} - \alpha_\ell A\overline{\mathbf{r}}^{\ell}_{AAR}, A\overline{\mathbf{r}}^{\ell}_{AAR}\rangle = 0,$$

which leads to

$$\begin{aligned}\|\overline{\mathbf{r}}^{\ell+m}_{AAR}\|_2^2 &= \langle \overline{\mathbf{r}}^{\ell}_{AAR} - \alpha_\ell A\overline{\mathbf{r}}^{\ell}_{AAR}, \overline{\mathbf{r}}^{\ell}_{AAR}\rangle \\ &= \langle \overline{\mathbf{r}}^{\ell}_{AAR}, \overline{\mathbf{r}}^{\ell}_{AAR}\rangle - \alpha_\ell\langle A\overline{\mathbf{r}}^{\ell}_{AAR}, \overline{\mathbf{r}}^{\ell}_{AAR}\rangle \\ &= \|\overline{\mathbf{r}}^{\ell}_{AAR}\|_2^2\left(1 - \frac{\langle A\overline{\mathbf{r}}^{\ell}_{AAR}, \overline{\mathbf{r}}^{\ell}_{AAR}\rangle}{\langle \overline{\mathbf{r}}^{\ell}_{AAR}, \overline{\mathbf{r}}^{\ell}_{AAR}\rangle}\frac{\langle A\overline{\mathbf{r}}^{\ell}_{AAR}, \overline{\mathbf{r}}^{\ell}_{AAR}\rangle}{\langle A\overline{\mathbf{r}}^{\ell}_{AAR}, A\overline{\mathbf{r}}^{\ell}_{AAR}\rangle}\right) \\ &= \|\overline{\mathbf{r}}^{\ell}_{AAR}\|_2^2\left(1 - \frac{\langle A\overline{\mathbf{r}}^{\ell}_{AAR}, \overline{\mathbf{r}}^{\ell}_{AAR}\rangle^2}{\langle \overline{\mathbf{r}}^{\ell}_{AAR}, \overline{\mathbf{r}}^{\ell}_{AAR}\rangle^2}\frac{\|\overline{\mathbf{r}}^{\ell}_{AAR}\|_2^2}{\|A\overline{\mathbf{r}}^{\ell}_{AAR}\|_2^2}\right).\end{aligned}$$

Note that

$$\frac{\langle A\mathbf{x}, \mathbf{x}\rangle}{\langle \mathbf{x}, \mathbf{x}\rangle} \geq \mu > 0, \quad \forall \mathbf{x} \in \mathbb{R}^n$$

and

$$\|A\overline{\mathbf{r}}^{\ell}_{AAR}\|_2 \leq \|A\|_2\|\mathbf{r}^{\ell}_{AAR}\|_2.$$

This guarantees a monotonic decrease of the residual norms:

$$\|\overline{\mathbf{r}}^{\ell+m}_{AAR}\|_2 \leq \left(1 - \frac{\mu^2}{\sigma^2}\right)^{\frac{1}{2}}\|\overline{\mathbf{r}}^{\ell}_{AAR}\|_2,$$

and Augmented AAR converges regardless of the initial guess. □

3.3 Numerical experiments

In this section we present some experiments to illustrate some of the asserts of the previous sections or to assess the performance of the schemes described. We employ a set of matrices selected from the SuiteSparse Matrix Collection (formerly known as the University of Florida Sparse Matrix Collection) [18], the Matrix Market Collection [65] and a few others given by collaborators at Oak Ridge National Laboratory (matrices `sp1`, `sp3`, `sp5` which arise from the numerical solution of radiation transport problems). In Table 3.1, we report the matrices and their most significant properties.

The performance of the linear solvers is also evaluated through *performance profiles* [19]. These graphic tools provide an immediate visual approach to compare the performance of multiple algorithms tested on a set of benchmark problems. In order to explain how this performance profiles are generated, let us refer to $\mathcal{S}$ as the set of solvers and $\mathcal{P}$ as the test set. We assume that we have n_s solvers and n_p problems. In this paper, performance profiles are used to compare the computational times. To this end we introduce

$$t_{p,s} = \text{computing time to solve problem } p \text{ with solver } s.$$

The comparison between the times taken by each solver is based on the performance ratio defined as

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in \mathcal{S}\}}.$$

The performance ratio allows one to compare the performance of solver s on problem p with the best performance by any solver to address the same problem p. In case a specific solver s does not succeed in solving problem p, then a convention is adopted to set $r_{p,s} = r_M$ where r_M is a maximal value. In our case we set $r_M = 10,000$. The performance of one solver compared to the others' on the whole benchmark set is displayed by the cumulative distribution function $\rho_s(\tau)$ which is defined as follows:

$$\rho_s(\tau) = \frac{1}{n_p}\text{size}\{p \in \mathcal{P} : r_{p,s} \leq \tau\}.$$

The value $\rho_s(\tau)$ represents the probability that solvers $s \in \mathcal{S}$ has a performance ratio $r_{p,s}$ less than or equal to the best possible ratio up to a scaling factor τ. The cumulative distribution function is nondecreasing, piecewise constant and continuous from the right at each discontinuity. A particular interpretation is associated with the value $\rho_s(1)$. In fact this value represents the probability that solver s outperforms every other solver from set $\mathcal{S}$ in solving a generic problem from set $\mathcal{P}$. The convention adopted that prescribes $r_{p,s} = r_M$ if solver s does not solve problem p leads to the reasonable assumption that

$$r_{p,s} \in [1, r_M].$$

Therefore, $\rho_s(r_M) = 1$ and

$$\rho_s^* = \lim_{\tau \to r_M^-} \rho_s(\tau)$$

represents the probability that solver s succeed in solving a generic problem from set $\mathcal{P}$. In general, solvers with larger values of $\rho_s(\tau)$ are to be preferred. It may happen that some solvers from set $\mathcal{S}$ take a considerable amount of time in solving specific problems from set $\mathcal{P}$. This consequently requires to pick a sufficiently high value for r_M and to extend the range of values of τ displayed in the performance profiles. This may lead to graphs that are difficult to interpret, since most of the main features of the curves may be shrunk on the far left of the graph window. Moreover, most of the window may be occupied to describe the trend of the curves for high values of τ

where nothing relevant happens. This motivates the replacement of $\rho_s(\tau)$ with

$$\tau \mapsto \frac{1}{n_p} \text{size}\{p \in \mathcal{P} : \log_2(r_{p,s}) \leq \tau\}.$$

Although using the log scale complicates the interpretation of the graph, it dedicates most of the space to values of τ where significant trends are captured and worth being discussed. From now on we use this quantity for each performance profile displayed.

3.3.1 Software and hardware configuration

The numerical results have been produced using the mathematical software MATLABR2016B with serial execution and without multithreading. The computer used is an Intel Xeon E5-4627.

3.3.2 Stagnation of AAR

A linear system on which both Full AR and Full AAR with $m < n$ stagnate without ever converging is given by the cyclic permutation matrix A and the right hand side $\mathbf{b}$ in $\mathbb{R}^n$ described as follows

$$A = \begin{bmatrix} 0 & 0 & 0 & \cdots & \cdots & 1 \\ 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \ddots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 & 0 \\ 0 & \cdots & 0 & \cdots & 1 & 0 \end{bmatrix}, \quad \mathbf{b} = \mathbf{e}_n = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}.$$

In fact Full GMRES stagnates for $n-1$ iterations and it eventually converges at the nth iteration. As concerns Full AR instead, the occurrence of a stagnation causes

Matrix	**Type**	**Size**	**Structure**	**Positive definite**	**Field of application**
1138_bus	real	1,138	symmetric	yes	power network
bcsstk08	real	1,074	symmetric	yes	structural engineering
bcsstk09	real	1,083	symmetric	yes	structural engineering
bcsstk10	real	1,086	symmetric	yes	structural engineering
crystm01	real	4,875	symmetric	yes	materials science
crystm02	real	13,965	symmetric	yes	materials science
nasa2910	real	2,910	symmetric	yes	structural engineering
raefsky4	real	19,779	symmetric	yes	structural engineering
msc01050	real	1,050	symmetric	yes	structural engineering
msc01440	real	1,440	symmetric	yes	structural engineering
msc23052	real	23,052	symmetric	yes	structural engineering
pwtk	real	217,918	symmetric	yes	structural engineering
ex10	real	2,410	symmetric	yes	computational fluid dynamics
ex10hs	real	2,548	symmetric	yes	computational fluid dynamics
ex15	real	6,867	symmetric	yes	computational fluid dynamics
cfd1	real	70,656	symmetric	yes	computational fluid dynamics
cfd2	real	123,440	symmetric	yes	computational fluid dynamics
qa8fm	real	66,127	symmetric	yes	acoustics
nos3	real	960	symmetric	yes	structural engineering
fidap029	real	2,870	nonsymmetric	yes	computational fluid dynamics
fidapm37	real	9,152	nonsymmetric	yes	computational fluid dynamics
raefsky5	real	6,316	nonsymmetric	yes	structural engineering
sp1	real	18,207	nonsymmetric	yes	nuclear engineering
sp3	real	36,414	nonsymmetric	yes	nuclear engineering
sp5	real	54,621	nonsymmetric	yes	nuclear engineering
bcsstk29	real	13,992	symmetric	no	structural engineering
sherman3	real	5,005	nonsymmetric	no	computational fluid dynamics
sherman5	real	3,312	nonsymmetric	no	computational fluid dynamics
fidap008	real	3,096	nonsymmetric	no	computational fluid dynamics
chipcool0	real	20,082	nonsymmetric	no	model reduction problem
e20r0000	real	4,241	nonsymmetric	no	computational fluid dynamics
spmsrtls	real	29,995	nonsymmetric	no	statistics/mathematics
III_Stokes	real	20,896	nonsymmetric	no	computational fluid dynamics
garon1	real	3,175	nonsymmetric	no	computational fluid dynamics
garon2	real	13,535	nonsymmetric	no	computational fluid dynamics
memplus	real	17,758	nonsymmetric	no	electronic circuit design
saylr4	real	3,564	nonsymmetric	no	oil reservoir modeling
xenon1	real	48,600	nonsymmetric	no	materials science
xenon2	real	157,464	nonsymmetric	no	materials science
venkat01	real	62,424	nonsymmetric	no	computational fluid dynamics
QC2534	complex	2,534	non-Hermitian	no	electromagnetics
mplate	complex	5,962	non-Hermitian	no	acoustic science
waveguide3	complex	21,306	non-Hermitian	no	electromagnetics
ABACUS_shell_hd	complex	23,412	non-Hermitian	no	model reduction
light_in_tissue	complex	29,282	non-Hermitian	no	electromagnetrics
kim1	complex	38,415	non-Hermitian	no	2D/3D problem
chevron2	complex	90,249	non-Hermitian	no	2D/3D problem

Table 3.1: AAR: list of matrices used for numerical experiments in MATLAB.

the algorithm to stagnate without ever recovering. The same pathological behavior is detected also for Full AAR if m is smaller than n. The only way to prevent stagnation in Full AAR on this problem requires setting m equal to n. However, this implies solving a least-squares problem where the matrix is full and of size $n \times n$. Therefore, the computational cost of the whole procedure for Full AAR would be higher than to solve the original linear system with Full GMRES.

3.3.3 Experiments to support Theorem 3.2.2

In this section we consider a linear system built on purpose to create a situation where Full GMRES periodically stagnates. The goal is to show that Full AAR is robust against the stagnation in this condition if a proper value of m is chosen, differently from Full AR that does not converge.

For a specified value b, we build a block-diagonal system with b diagonal blocks, the kth of which is a $2k \times 2k$ circulant matrix. The right hand side is constructed so that , with zero initial guess, Full GMRES makes progress on even-numbered iterations and stagnates on odd-numbered ones. The interval of stagnation in this case has length equal to one. Full AR never converges in this situation, whereas Full AAR with $m > 1$ attains the prescribed accuracy in a finite number of iterations.

Changing the length of the stagnation, say ℓ, requires building diagonal blocks of size $\ell k \times \ell k$. For different choices of the value ℓ to tune the length of the stagnation, we verified that Full AR never converges whereas Full AAR with $m > \ell$ always attains the prescribed accuracy in a finite number of iterations.

3.3.4 Error mode analysis for AAR

Most of this work has been focused on studying analogies between Anderson-accelerated Richardson schemes and GMRES. However, because of the multilevel nature of both AR and AAR, it is possible to find analogies with multigrid methods [13]. Indeed, the alternation of Richardson sweeps and Anderson mixing appears to efficiently damp high frequency error components with the former and low frequency error components with the latter. In order to investigate this property, we consider a 1D Laplace problem on the interval $(0, 1)$ with homogeneous Dirichlet boundary conditions. The operator is discretized using second order centered finite differences with 100 internal nodes. We solve the 100×100 linear system with AAR, setting the Richardson relaxation parameter $\omega = 0.5$, the Anderson mixing weights $\beta_k = 1$, the history length $p = 10$, the length of the periodic interval for relaxation sweeps $m = 6$ and no preconditioning. The behavior of the error is monitored across several iterations. The first six steps of the Jacobi relaxation damp efficiently the components of the error associated with high frequencies, leaving the low frequency components mostly unchanged. In Figure 3.2 we show the error behavior from iteration 7 through iteration 13. It is noticed that the Richardson relaxation cannot operate efficiently on the remaining frequencies of the error which are very smooth. In fact, smooth components of the error are associated with low frequencies. However, the Anderson mixing significantly contributes in modifying the error profile when the Anderson acceleration is applied at iteration 13. This confirms that the Anderson acceleration effectively damps also low frequencies of the error. Therefore, the least-squares problem operates similarly to a coarse grid correction in the multigrid framework, lowering components of the error that a simple one level fixed point scheme cannot efficiently treat. However, saying that the Jacobi preconditioner and the Anderson mixing address separate components of the error is not accurate. In Figure 3.3 we show the error plot across the discretized domain at the sixth and seventh iteration. The error curve at the seventh iteration suggests that the Anderson mixing has also regularizing (or smoothing) properties on high frequency error components. In fact, high frequencies of the error are efficiently smoothed from iteration 6 to iteration 7. Therefore, Anderson acceleration can damp error components associated with the whole spectrum.

3.3.5 Comparison between Truncated AR and Truncated AAR

In order to illustrate the advantages of periodic alternation between standard Richardson sweeps and least-squares problem, we provide here results comparing the performance of Truncated AR versus Truncated AAR. The check has been conducted without preconditioner, with the diagonal preconditioner, ILU(0) and ILUT(10^{-4}) or IC(0) and ICT(10^{-4}) for symmetric positive definite matrices.

Results without preconditioner are reported in Table 3.2. For most of the test cases Truncated AAR(6,10) outperforms Truncated AR(10) in terms of computational time to reach a prescribed accuracy. The only exceptions are the matrices `e20r0000`, `garon1` and `garon2`. A possible explanation to this exceptions may be the inefficacy of Richardson sweeps in Truncated AAR(6,10) to generate a proper projection subspace, whereas solving a least-square problem at each iteration may facilitate Truncated AR(10) to construct a more efficient subspace. Moreover, Truncated AAR(6,10) succeeds in converging for some numerical tests where Truncated AR(10) never converges, such as for the matrices `pwtk`, `sherman5`, `ABACUS_shell_hd` and `chevron2`. The conclusion about Truncated AAR(6,10) converging faster than Truncated AR(10) can also be drawn via the performance profile in Figure 3.4. In fact, the continuous line associated with Trunctaed AAR(6,10) is always higher than the dashed line for Truncated AR(10).

Results for the use of a diagonal preconditioner are reported in Table 3.3. The diagonal preconditioner is very simple to apply but does not always decrease the computational time with respect to no use of preconditioners. However, Truncated AAR(6,10) still performs better Truncated AR(10), as it is also confirmed by the performance profile in Figure 3.5.

Numerical results when ILU(0) or IC(0) is used as a preconditioner are displayed in Table 3.4. Results are shown only for those cases that allow a stable construction of the preconditioner. We highlight the fact that Truncated AR(10) succeeds in converging on linear systems with the matrix `mplate`, whereas Truncated AAR(6,10) does not. The difference between the timings spent by Truncated AR(10) and Truncated AAR(6,10) is less pronounced in this case because the quality of P^{-1} facilitates the convergence of the linear solver, albeit Truncated AAR(6,10) generally converges in less time than Truncated AR(10) in this case too. The fact that the performances between the two linear solvers are not neatly separated as before is also visible in the performance profile in Figure 3.6. Indeed, the two curves eventually overlap on the right side of the graph.

The results for the the use of ILUT(10^{-4}) or ICT(10^{-4}) are shown in Table 3.5. In this case as well, Truncated AAR(6,10) outperforms Truncated AR(10), as confirmed by the performance profile in Figure 3.7.

Summarizing the results for different problems and different preconditioners, we can conclude that inserting multiple relaxation steps between consecutive least-squares problems significantly decreases the time to compute a solution to given accuracy.

3.3.6 Comparison between Preconditioned Conjugate Gradient and Truncated AAR

In this section we compare the performance of Truncated AAR on symmetric positive definite matrices with the (preconditioned) conjugate gradient (PCG) method [36]. Tests have been run without preconditioner, with a diagonal preconditioner, with IC(0) and with ICT(10^{-4}). As concerns the use of IC(0) or ICT(10^{-4}), simulations have been run only on cases that allow a numerically stable construction of the preconditioner. Results in Table 3.6 correspond to no use of the preconditioner. Although there is no clear evidence of a method prevailing over the other for all the problems considered, the performance profile in Figure 3.8 shows that CG performs better than Truncated AAR(6,10) overall. This can be explained in part with the different principles adopted by the two algorithms to update the approximate solution. In fact,

	Truncated AR$(p = 10)$			**Truncated AAR**$(m = 6,\ p = 10)$		
Matrix	Time (s)	rel. error	# Itr.	Time (s)	rel. error	# Itr.
1138_bus	1.63	$2.72 \cdot 10^{-3}$	3,565	0.96	$2.25 \cdot 10^{-3}$	2,992
bcsstk08	3.73	$4.11 \cdot 10^{-3}$	11,008	0.19	$6.51 \cdot 10^{-5}$	395
bcsstk09	0.92	$2.17 \cdot 10^{-5}$	2,579	0.20	$9.25 \cdot 10^{-6}$	352
bcsstk10	7.18	$3.26 \cdot 10^{-4}$	2,154	0.94	$2.95 \cdot 10^{-4}$	3,008
crystm01	0.29	$1.10 \cdot 10^{-6}$	191	0.12	$4.36 \cdot 10^{-7}$	103
crystm02	54.09	0.01	10,348	11.5	0.01	2,699
nasa2910	63.35	$1.05 \cdot 10^{-3}$	6,995	3.89	$7.94 \cdot 10^{-4}$	6,072
raefsky4	978.60	0.20	124,940	217.69	0.19	40,829
msc01050	33.07	0.03	1,009	3.17	0.04	16,736
msc01440	41.08	$2.06 \cdot 10^{-3}$	88,922	1.84	$1.83 \cdot 10^{-3}$	5,458
msc23052	265.16	0.12	30,748	110.47	0.12	20,245
pwtk	-	-	-	1,415.02	0.02	22,813
ex10	15.97	0.37	23,635	13.13	0.42	6,042
ex10hs	12.18	0.40	17,019	1.91	0.45	4,144
ex15	22.70	0.57	13,121	2.33	0.57	2,000
cfd1	981.88	$3.01 \cdot 10^{-4}$	31,731	18.30	$1.79 \cdot 10^{-4}$	1,222
cfd2	465.68	$1.70 \cdot 10^{-4}$	9,246	61.66	$7.13 \cdot 10^{-4}$	2,377
qa8fm	2.13	$4.16 \cdot 10^{-7}$	78	1.14	$1.58 \cdot 10^{-7}$	67
fidap029	0.22	$1.45 \cdot 10^{-6}$	146	0.12	$7.89 \cdot 10^{-7}$	124
fidapm37	559.72	0.04	151,540	176.01	0.04	16,464
raefsky5	1.66	0.68	1,142	0.21	0.64	205
sp1	1.86	$3.05 \cdot 10^{-7}$	291	0.52	$2.05 \cdot 10^{-7}$	157
sp3	4.08	$2.61 \cdot 10^{-7}$	305	1.06	$2.08 \cdot 10^{-7}$	179
sp5	5.84	$2.37 \cdot 10^{-7}$	295	1.67	$1.90 \cdot 10^{-7}$	173
bcsstk29	-	-	-	-	-	-
sherman3	21.01	0.46	15,865	5.33	0.46	7,202
sherman5	-	-	-	4.37	$1.58 \cdot 10^{-5}$	8,057
fidap008	401.87	0.12	387,130	3.65	0.09	6,487
chipcool0	170.50	$3.32 \cdot 10^{-3}$	28,168	15.40	$3.03 \cdot 10^{-3}$	4,774
e20r0000	8.42	$7.45 \cdot 10^{-3}$	6,579	199.13	$1.25 \cdot 10^{-3}$	195,460
spmsrtls	-	-	-	-	-	-
III_Stokes	91.26	0.09	13,636	20.59	0.09	6,570
garon1	19.23	$5.46 \cdot 10^{-3}$	21,501	59.13	$2.71 \cdot 10^{-3}$	106,900
garon2	169.54	$8.30 \cdot 10^{-3}$	38,114	747.91	$7.44 \cdot 10^{-3}$	222,743
memplus	8.12	$1.88 \cdot 10^{-5}$	1,464	3.53	$9.84 \cdot 10^{-6}$	1,366
saylr4	42.34	$8.89 \cdot 10^{-3}$	40,596	1.25	$9.16 \cdot 10^{-3}$	2,025
xenon1	52.09	$4.64 \cdot 10^{-5}$	2,998	6.06	$4.66 \cdot 10^{-5}$	700
xenon2	140.28	$4.95 \cdot 10^{-5}$	2,405	23.55	$6.97 \cdot 10^{-5}$	734
venkat01	-	-	-	-	-	-
QC2534	-	-	-	-	-	-
mplate	-	-	-	-	-	-
waveguide3	903.28	$1.60 \cdot 10^{-5}$	51,060	93.10	$1.18 \cdot 10^{-5}$	10,587
ABACUS_shell_hd	-	-	-	844.82	0.68	91,130
light_in_tissue	50.75	$1.63 \cdot 10^{-5}$	2,348	4.22	$1.10 \cdot 10^{-5}$	379
kim1	3,258.94	$1.38 \cdot 10^{-4}$	87,439	92.74	$1.24 \cdot 10^{-5}$	5,476
chevron2	-	-	-	3,135.26	$1.57 \cdot 10^{-5}$	46,917

Table 3.2: Truncated AR(10) vs. Truncated AAR(6,10). Experiments without preconditioner.

	Truncated AR($p = 10$)			**Truncated AAR**($m = 6$, $p = 10$)		
Matrix	Time (s)	rel. error	# Itr.	Time (s)	rel. error	# Itr.
1138_bus	33.65	$4.01 \cdot 10^{-5}$	76,618	3.66	$9.80 \cdot 10^{-4}$	20,802
bcsstk08	0.69	$6.82 \cdot 10^{-5}$	1,810	0.19	$6.51 \cdot 10^{-5}$	395
bcsstk09	0.47	$3.76 \cdot 10^{-5}$	1,089	0.19	$4.14 \cdot 10^{-5}$	352
bcsstk10	0.36	$2.68 \cdot 10^{-5}$	762	0.27	$2.46 \cdot 10^{-5}$	665
crystm01	0.16	$1.58 \cdot 10^{-7}$	51	0.06	$5.17 \cdot 10^{-8}$	49
crystm02	34.84	0.02	5,624	7.02	0.01	2,240
nasa2910	2.13	$4.66 \cdot 10^{-4}$	2,123	1.11	$5.62 \cdot 10^{-4}$	1,798
raefsky4	-	-	-	8.33	0.11	169
msc01050	10.41	0.21	30,815	5.64	0.20	22,695
msc01440	6.34	$5.08 \cdot 10^{-5}$	13,931	0.39	$4.92 \cdot 10^{-5}$	753
msc23052	-	-	-	104.07	0.16	13,372
pwtk	-	-	-	114.27	0.04	1,847
ex10	-	-	-	92.75	0.35	202,890
ex10hs	-	-	-	77.55	0.34	161,780
ex15	-	-	-	243.36	0.91	185,360
cfd1	-	-	-	17.41	$2.32 \cdot 10^{-4}$	1,309
cfd2	-	-	-	61.44	$8.51 \cdot 10^{-4}$	2,265
qa8fm	1.89	$1.42 \cdot 10^{-7}$	49	0.82	$3.67 \cdot 10^{-8}$	49
fidap029	0.07	$5 \cdot 10^{-9}$	26	0.05	$8.07 \cdot 10^{-7}$	26
fidapm37	106.69	0.01	31,432	11.93	0.01	7,075
raefsky5	0.14	$9.98 \cdot 10^{-9}$	24	0.08	$7.19 \cdot 10^{-9}$	25
sp1	1.05	$1.60 \cdot 10^{-7}$	143	0.46	$2.39 \cdot 10^{-7}$	120
sp3	2.06	$2.70 \cdot 10^{-7}$	153	0.86	$3.16 \cdot 10^{-7}$	129
sp5	3.15	$2.06 \cdot 10^{-7}$	163	1.50	$3.04 \cdot 10^{-7}$	137
bcsstk29	-	-	-	-	-	-
sherman3	5.08	$1.10 \cdot 10^{-4}$	2,944	1.40	$1.12 \cdot 10^{-4}$	1,346
sherman5	0.42	$1.64 \cdot 10^{-6}$	308	0.24	$1.57 \cdot 10^{-6}$	261
fidap008	-	-	-	71.57	0.06	129,410
chipcool0	5.12	$6.10 \cdot 10^{-7}$	770	1.67	$5.92 \cdot 10^{-7}$	506
e20r0000	104.39	$3 \cdot 10^{-3}$	84,627	4.09	$2.70 \cdot 10^{-3}$	6,988
spmsrtls	-	-	-	-	-	-
III_Stokes	67.98	0.09	11,741	17.15	0.09	6,256
garon1	160.65	$1.15 \cdot 10^{-3}$	183,129	8.07	$1.05 \cdot 10^{-3}$	18,196
garon2	-	-	-	47.18	$4.31 \cdot 10^{-3}$	27,195
memplus	29.98	$3.51 \cdot 10^{-5}$	1,678	1.03	$3.66 \cdot 10^{-5}$	380
saylr4	8.45	$9.56 \cdot 10^{-3}$	7,850	6.21	$9.59 \cdot 10^{-3}$	1,684
xenon1	24.45	$2.32 \cdot 10^{-5}$	1,581	6.85	$2.55 \cdot 10^{-5}$	781
xenon2	82.12	$8.02 \cdot 10^{-5}$	1,436	26.88	$6.58 \cdot 10^{-5}$	818
venkat01	-	-	-	1,352.34	$2.43 \cdot 10^{-4}$	78,742
QC2534	-	-	-	-	-	-
mplate	-	-	-	-	-	-
waveguide3	1,299.81	$5.58 \cdot 10^{-5}$	7,540	138.09	$9.95 \cdot 10^{-6}$	14,134
ABACUS_shell_hd	-	-	-	-	-	-
light_in_tissue	11.75	$1.31 \cdot 10^{-5}$	449	5.08	$5.21 \cdot 10^{-6}$	361
kim1	4.95	$3.37 \cdot 10^{-7}$	158	2.49	$3.55 \cdot 10^{-7}$	145
chevron2	318.79	$1.16 \cdot 10^{-5}$	4,241	275.81	$9.14 \cdot 10^{-6}$	6,783

Table 3.3: Truncated AR(10) vs. Truncated AAR(6,10). Experiments with diagonal preconditioner.

	Truncated AR($p = 10$)			**Truncated AAR**($m = 6$, $p = 10$)		
Matrix	Time (s)	rel. error	# Itr.	Time (s)	rel. error	# Itr.
1138_bus	0.99	$7.44 \cdot 10^{-6}$	1,868	0.84	$7.24 \cdot 10^{-6}$	2,508
bcsstk08	0.08	$7.08 \cdot 10^{-7}$	54	0.07	$3.46 \cdot 10^{-7}$	53
bcsstk09*	$*$	$*$	$*$	$*$	$*$	$*$
bcsstk10*	$*$	$*$	$*$	$*$	$*$	$*$
crystm01	0.04	$7.24 \cdot 10^{-9}$	4	0.04	$1.38 \cdot 10^{-12}$	7
crystm02	34.22	0.02	5,429	3.94	0.02	794
nasa2910*	$*$	$*$	$*$	$*$	$*$	$*$
raefsky4*	$*$	$*$	$*$	$*$	$*$	$*$
msc01050*	$*$	$*$	$*$	$*$	$*$	$*$
msc01440*	$*$	$*$	$*$	$*$	$*$	$*$
msc23052*	$*$	$*$	$*$	$*$	$*$	$*$
pwtk*	$*$	$*$	$*$	$*$	$*$	$*$
ex10*	$*$	$*$	$*$	$*$	$*$	$*$
ex10hs	$*$	$*$	$*$	$*$	$*$	$*$
ex15	$*$	$*$	$*$	$*$	$*$	$*$
cfd1	17,48	$4.23 \cdot 10^{-5}$	658	3.94	$5.13 \cdot 10^{-5}$	538
cfd2*	$*$	$*$	$*$	$*$	$*$	$*$
qa8fm	0.27	$1.43 \cdot 10^{-9}$	10	0.34	$9.41 \cdot 10^{-8}$	13
fidap029	0.05	$8 \cdot 10^{-11}$	8	0.01	$4.60 \cdot 10^{-13}$	19
fidapm37*	$*$	$*$	$*$	$*$	$*$	$*$
raefsky5	0.04	$2.55 \cdot 10^{-11}$	6	0.05	$2.05 \cdot 10^{-10}$	7
sp1	0.38	$3.05 \cdot 10^{-8}$	44	0.29	$3.51 \cdot 10^{-8}$	53
sp3	0.72	$5.61 \cdot 10^{-8}$	46	0.57	$2.97 \cdot 10^{-8}$	43
sp5	1.04	$4.94 \cdot 10^{-8}$	45	0.72	$2.72 \cdot 10^{-8}$	43
bcsstk29*	$*$	$*$	$*$	$*$	$*$	$*$
sherman3	0.76	$4.13 \cdot 10^{-6}$	339	0.38	$4.28 \cdot 10^{-9}$	263
sherman5	0.17	$3.40 \cdot 10^{-7}$	83	0.23	$1.88 \cdot 10^{-7}$	61
fidap008*	-	-	-	-	-	-
chipcool0	1.42	$2.2 \cdot 10^{-7}$	183	0.52	$8.43 \cdot 10^{-8}$	117
e20r0000*	$*$	$*$	$*$	$*$	$*$	$*$
spmsrtls	0.07	$7.65 \cdot 10^{-14}$	3	0.09	$1.69 \cdot 10^{-13}$	7
III_Stokes*	$*$	$*$	$*$	$*$	$*$	$*$
garon1*	$*$	$*$	$*$	$*$	$*$	$*$
garon2*	$*$	$*$	$*$	$*$	$*$	$*$
memplus	3.98	$9.2 \cdot 10^{-6}$	720	1.02	$1.22 \cdot 10^{-5}$	360
saylr4	0.24	$5.5 \cdot 10^{-9}$	146	0.15	$7.98 \cdot 10^{-9}$	123
xenon1*	-	-	-	-	-	-
xenon2*	-	-	-	-	-	-
venkat01	12.05	$7.82 \cdot 10^{-7}$	470	7.94	$7.98 \cdot 10^{-7}$	379
QC2534	151.11	$1.01 \cdot 10^{-5}$	31,426	350.57	$9.58 \cdot 10^{-3}$	90,685
mplate	349.96	0.11	73,701	-	-	-
waveguide3	-	-	-	-	-	-
ABACUS_shell_hd	-	-	-	-	-	-
light_in_tissue	4.57	$9.15 \cdot 10^{-7}$	142	1.73	$4.86 \cdot 10^{-7}$	114
kim1	0.47	$1.78 \cdot 10^{-9}$	13	0.33	$7.72 \cdot 10^{-9}$	14
chevron2	161.55	$9.66 \cdot 10^{-7}$	1,385	44.88	$9.30 \cdot 10^{-7}$	867

Table 3.4: Truncated AR(10) vs. Truncated AAR(6,10). Experiments with IC(0) preconditioner for real symmetric positive definite matrices and ILU(0) preconditioner for real matrices that are not symmetric positive definite.

	Truncated AR($p = 10$)			**Truncated AAR**($m = 6$, $p = 10$)		
Matrix	Time (s)	rel. error	# Itr.	Time (s)	rel. error	# Itr.
1138_bus	0.06	$2.71 \cdot 10^{-8}$	26	0.04	$3.34 \cdot 10^{-8}$	25
bcsstk08*	0.07	$2.22 \cdot 10^{-8}$	13	0.04	$1.93 \cdot 10^{-8}$	13
bcsstk09	0.06	$6.16 \cdot 10^{-9}$	9	0.06	$6.16 \cdot 10^{-9}$	9
bcsstk10	0.07	$5.72 \cdot 10^{-9}$	11	0.04	$5.73 \cdot 10^{-10}$	13
crystm01	0.06	$1.94 \cdot 10^{-11}$	5	0.04	$1.52 \cdot 10^{-11}$	7
crystm02*	*	*	*	*	*	*
nasa2910*	0.09	$5.13 \cdot 10^{-8}$	29	0.08	$2.64 \cdot 10^{-8}$	30
raefsky4*	121.41	$1.04 \cdot 10^{-5}$	5,410	108.64	$1.04 \cdot 10^{-5}$	5,410
msc01050	0.06	0.1	16	0.04	0.10	19
msc01440	0.09	$4.64 \cdot 10^{-7}$	35	0.04	0.10	19
msc23052*	*	*	*	*	*	*
pwtk*	*	*	*	*	*	*
ex10*	*	*	*	*	*	*
ex10hs*	*	*	*	*	*	*
ex15*	*	*	*	*	*	*
cfd1*	*	*	*	*	*	*
cfd2*	*	*	*	*	*	*
qa8fm	0.12	$1.61 \cdot 10^{-11}$	5	0.14	$1.28 \cdot 10^{-11}$	7
fidap029	0.07	$9.58 \cdot 10^{-12}$	5	0.06	$7.48 \cdot 10^{-12}$	7
fidapm37*	*	*	*	*	*	*
rafesky5	0.06	$8.60 \cdot 10^{-11}$	5	0.04	$1.92 \cdot 10^{-11}$	7
sp1	0.11	$3.97 \cdot 10^{-12}$	6	0.13	$1.20 \cdot 10^{-9}$	7
sp3	0.18	$2.28 \cdot 10^{-9}$	7	0.16	$2.14 \cdot 10^{-9}$	7
sp5	0.26	$2.89 \cdot 10^{-9}$	7	0.22	$6.85 \cdot 10^{-9}$	8
bcsstk29*	*	*	*	*	*	*
sherman3	0.10	$5.23 \cdot 10^{-9}$	11	0.10	$2.52 \cdot 10^{-10}$	13
sherman5	0.08	$1.05 \cdot 10^{-9}$	13	0.08	$1.84 \cdot 10^{-9}$	13
fidap008	-	-	-	-	-	-
chipcool0	0.28	$3.59 \cdot 10^{-9}$	14	0.29	$6.72 \cdot 10^{-9}$	16
e20r0000	0.16	$4.65 \cdot 10^{-6}$	25	0.16	$1.11 \cdot 10^{-5}$	27
spmsrtls	0.07	$1.38 \cdot 10^{-10}$	5	0.07	$5.24 \cdot 10^{-11}$	7
III_Stokes	0.50	$8.44 \cdot 10^{-5}$	24	0.60	$8.34 \cdot 10^{-5}$	36
garon1	0.14	$2.18 \cdot 10^{-8}$	22	0.11	$7.24 \cdot 10^{-8}$	22
garon2	0.82	$1.26 \cdot 10^{-8}$	35	0.82	$1.36 \cdot 10^{-7}$	37
memplus	0.12	$5.77 \cdot 10^{-9}$	11	0.09	$4.05 \cdot 10^{-10}$	13
saylr4	0.28	$1.23 \cdot 10^{-6}$	173	0.13	$1.35 \cdot 10^{-6}$	107
xenon1	1.01	$9.10 \cdot 10^{-9}$	26	0.91	$9.16 \cdot 10^{-10}$	25
xenon2	4.31	$1.87 \cdot 10^{-8}$	29	2.99	$1.95 \cdot 10^{-8}$	27
venkat01	1.64	$4.96 \cdot 10^{-8}$	15	1.94	$3.17 \cdot 10^{-10}$	19
QC2534	0.10	$5.41 \cdot 10^{-9}$	9	0.14	$1.11 \cdot 10^{-11}$	13
mplate	56.62	0.02	4,566	16.41	0.04	2,048
waveguide3	0.73	$6.23 \cdot 10^{-9}$	12	0.76	$6.16 \cdot 10^{-9}$	13
ABACUS_shell_hd	1.07	6.90	40	1.14	9.05	72
light_in_tissue	0.41	$3.31 \cdot 10^{-9}$	13	0.34	$7.31 \cdot 10^{-9}$	15
kim1	0.24	$3.87 \cdot 10^{-9}$	10	0.23	$3.87 \cdot 10^{-9}$	7
chevron2	6.39	$6.00 \cdot 10^{-8}$	57	4.56	$3.19 \cdot 10^{-8}$	61

Table 3.5: Truncated AR(10) vs. Truncated AAR(6,10). Experiments with ICT(10^{-4}) preconditioner for real symmetric positive definite matrices and ILUT(10^{-4}) preconditioner for real matrices that are not symmetric positive definite.

Truncated AAR(6,10) aims to reduce the ℓ^2-norm of the residual on a projection subspace, whereas CG aims to minimize the A-norm of the error. Therefore, the performances of the two algorithms are not entirely comparable, since they operate according to different optimality criteria. Moreover, the stopping criterion based on the relative residual ℓ^2-norm favors Truncated AAR over PCG. Regardless of this fact, the situation becomes more neat with the diagonal preconditioner. In fact numerical results in Table 3.7 clearly show PCG outperforming Truncated AAR(6,10). This is easily noticed also through the performance profile in Figure 3.9, where the curve of PCG is always above the curve of Truncated AAR(6,10). Similarly, results in Table 3.8 for the IC(0) preconditioner and in Table 3.9 for the ICT(10^{-4}) preconditioner favor PCG over Truncated AAR(6,10), as highlighted also by the performance profile in Figures 3.10 and 3.11. Although the number of iteration taken by Truncated AAR to converge may seem too high according to reasonable standards, we remind the reader that the computational cost and efficiency are not the same for each iteration. In fact, the least-squares problem is computed periodically after a batch of multiple Richardson iterations. The relaxation steps have the advantage of being easy to compute, but they do not necessarily minimize any quantity with respect to an optimality criterion. Therefore, it is reasonable to expect the number of iterations for AAR to be higher than for any standard Krylov method.

3.3.7 Comparison between Truncated AAR and Restarted GMRES

In this section we compare Truncated AAR with Restarted GMRES. The comparison is again carried out without preconditioners, with a diagonal preconditioner, with ILU(0) and with ILUT(10^{-4}). Results with ILU(0) and ILUT(10^{-4}) are reported only if the preconditioner can be constructed in a numerically stable way.

The length m of the cycle in Restarted GMRES is set to $m = 10$ and $m = 30$.

	CG			**Truncated AAR**$(m = 6,\, p = 10)$		
Matrix	Time (s)	rel.error	# Itr.	Time (s)	rel. error	# Itr.
1138_bus	0.42	$8.88 \cdot 10^{-3}$	2,058	0.96	$2.25 \cdot 10^{-3}$	2,992
bcsstk08	0.61	$7.70 \cdot 10^{-4}$	2,932	2.73	$7.21 \cdot 10^{-3}$	9,292
bcsstk09	0.07	$1.95 \cdot 10^{-7}$	271	0.20	$9.25 \cdot 10^{-6}$	352
bcsstk10	0.59	$2.21 \cdot 10^{-5}$	2,812	0.94	$2.95 \cdot 10^{-4}$	3,008
crystm01	0.78	$1.48 \cdot 10^{-7}$	94	0.12	$4.36 \cdot 10^{-7}$	103
crystm02	29.13	0.01	3,344	11.5	0.01	2,699
nasa2910	29.06	$3.18 \cdot 10^{-5}$	5,230	3.89	$7.94 \cdot 10^{-4}$	6,072
raefsky4	27.83	0.07	2,868	217.69	0.19	40,829
msc01050	1.27	0.02	5,738	3.17	0.04	16,736
msc01440	0.94	$1.68 \cdot 10^{-4}$	4,567	1.84	$1.83 \cdot 10^{-3}$	5,458
msc23052	1,678.6	0.05	156,313	110.47	0.12	20,245
pwtk	2,840.03	0.01	58,159	1,415.02	0.02	22,813
ex10	0.31	0.18	2.67	13.13	0.42	6,042
ex10hs	0.26	0.20	1,025	1.91	0.45	4,144
ex15	7,74	0.57	1,454	2.33	0.57	2,000
cfd1	26.37	$4.11 \cdot 10^{-6}$	1,417	18.30	$1.79 \cdot 10^{-4}$	1,222
cfd2	103.79	$3.15 \cdot 10^{-5}$	3,824	61.66	$7.13 \cdot 10^{-4}$	2,377
qa8fm	1.23	$1.29 \cdot 10^{-7}$	62	1.14	$1.58 \cdot 10^{-7}$	67

Table 3.6: CG vs. Truncated AAR(6,10) on real symmetric positive definite matrices. Experiments without preconditioner.

	PCG			**Truncated AAR**($m = 6$, $p = 10$)		
Matrix	Time (s)	rel.error	# Itr.	Time (s)	rel. error	# Itr.
1138_bus	0.15	$9.04 \cdot 10^{-6}$	883	3.66	$9.80 \cdot 10^{-4}$	20,802
bcsstk08	0.02	$1.67 \cdot 10^{-5}$	130	0.19	$6.51 \cdot 10^{-5}$	395
bcsstk09	0.05	$1.77 \cdot 10^{-7}$	249	0.19	$4.14 \cdot 10^{-5}$	352
bcsstk10	0.09	$1.35 \cdot 10^{-5}$	564	0.27	$2.46 \cdot 10^{-5}$	665
crystm01	0.15	$9.05 \cdot 10^{-8}$	41	0.06	$5.17 \cdot 10^{-8}$	49
crystm02	24.56	0.01	2,641	7.02	0.01	2,240
nasa2910	7.39	$1.43 \cdot 10^{-5}$	1,191	1.11	$5.62 \cdot 10^{-4}$	1,798
raefsky4	8.19	0.11	437	8.33	0.11	169
msc01050	0.13	0.203	556	5.64	0.20	22,697
msc01440	0.16	$2.61 \cdot 10^{-5}$	558	0.39	$4.92 \cdot 10^{-5}$	753
msc23052	123.92	0.03	12,162	104.07	0.16	13,372
pwtk	305.20	0.02	6,503	114.27	0.04	1,847
ex10	0.94	0.16	2,945	92.75	0.35	202,890
ex10hs	0.91	0.14	2,756	77.55	0.34	161,780
ex15	5.94	0.71	915	243.36	0.91	185,360
cfd1	29.82	$2.12 \cdot 10^{-6}$	1,470	17.41	$2.32 \cdot 10^{-4}$	1,309
cfd2	99.28	$3.46 \cdot 10^{-5}$	3,701	61.44	$8.51 \cdot 10^{-4}$	2,265
qa8fm	0.98	$6.73 \cdot 10^{-8}$	41	0.82	$3.67 \cdot 10^{-8}$	49

Table 3.7: PCG vs. Truncated AAR(6,10) on real symmetric positive definite matrices. Experiments with diagonal preconditioner.

	PCG			**Truncated AAR**$(m = 6,\, p = 10)$		
Matrix	Time (s)	rel.error	# Itr.	Time (s)	rel. error	# Itr.
1138_bus	0.03	$2.47 \cdot 10^{-6}$	128	0.84	$7.24 \cdot 10^{-6}$	2,508
bcsstk08	0.01	$7.39 \cdot 10^{-6}$	24	0.07	$3.46 \cdot 10^{-7}$	53
bcsstk09*	$*$	$*$	$*$	$*$	$*$	$*$
bcsstk10*	$*$	$*$	$*$	$*$	$*$	$*$
crystm01	0.03	$1.15 \cdot 10^{-8}$	2	0.04	$1.38 \cdot 10^{-12}$	7
crystm02	3.38	0.02	283	3.94	0.02	794
nasa2910*	$*$	$*$	$*$	$*$	$*$	$*$
raefsky4*	$*$	$*$	$*$	$*$	$*$	$*$
msc01050*	$*$	$*$	$*$	$*$	$*$	$*$
msc01440*	$*$	$*$	$*$	$*$	$*$	$*$
msc23052*	$*$	$*$	$*$	$*$	$*$	$*$
pwtk*	$*$	$*$	$*$	$*$	$*$	$*$
ex10*	$*$	$*$	$*$	$*$	$*$	$*$
ex10hs*	$*$	$*$	$*$	$*$	$*$	$*$
ex15	$*$	$*$	$*$	$*$	$*$	$*$
cfd1	8.84	$5.48 \cdot 10^{-6}$	402	9.37	$5.13 \cdot 10^{-5}$	538
cfd2*	$*$	$*$	$*$	$*$	$*$	$*$
qa8fm	0.24	$2.73 \cdot 10^{-8}$	6	0.34	$9.41 \cdot 10^{-8}$	13

Table 3.8: PCG vs. Truncated AAR(6,10) on real symmetric positive definite matrices. Experiments with IC(0) preconditioner.

	PCG			**Truncated AAR**$(m = 6,\, p = 10)$		
Matrix	Time (s)	rel.error	# Itr.	Time (s)	rel. error	# Itr.
1138_bus	0.01	$3.44 \cdot 10^{-5}$	12	0.05	$8.23 \cdot 10^{-8}$	26
bcsstk08	0.01	$7.99 \cdot 10^{-7}$	9	0.04	$2.22 \cdot 10^{-8}$	13
bcsstk09*	$*$	$*$	$*$	$*$	$*$	$*$
bcsstk10*	$*$	$*$	$*$	$*$	$*$	$*$
crystm01	0.02	$6.02 \cdot 10^{-8}$	2	0.06	$1.94 \cdot 10^{-11}$	5
crystm02*	$*$	$*$	$*$	$*$	$*$	$*$
nasa2910*	0.19	$1.44 \cdot 10^{-6}$	16	0.13	$5.13 \cdot 10^{-8}$	29
raefsky4	0.93	0.80	37	107.28	$1.04 \cdot 10^{-5}$	5,410
msc01050	0.01	0.10	9	0.06	0.10	16
msc01440	0.02	$6.29 \cdot 10^{-6}$	22	0.08	$4.64 \cdot 10^{-7}$	35
msc23052*	$*$	$*$	$*$	$*$	$*$	$*$
pwtk*	$*$	$*$	$*$	$*$	$*$	$*$
ex10*	$*$	$*$	$*$	$*$	$*$	$*$
ex10hs*	$*$	$*$	$*$	$*$	$*$	$*$
ex15	$*$	$*$	$*$	$*$	$*$	$*$
cfd1*	$*$	$*$	$*$	$*$	$*$	$*$
cfd2*	$*$	$*$	$*$	$*$	$*$	$*$
qa8fm	0.09	$3.24 \cdot 10^{-8}$	2	0.15	$1.61 \cdot 10^{-11}$	5

Table 3.9: PCG vs. Truncated AAR(6,10) on real symmetric positive definite matrices. Experiments with ICT(10^{-4}) preconditioner.

Different tables are displayed for each linear solver (Truncated AAR(6,10), Restarted GMRES(10) and Restarted GMRES(30)). Each table shows the averaged time, the averaged relative error attained, the averaged number of iterations and the value of the angles $\angle(\mathbf{r}^{k+m}, \mathbf{r}^k)$ and $\angle(\mathbf{r}^{k+2m}, \mathbf{r}^k)$. With regards to the angles, the averaging accounts both for the numbers of runs and the number of iterations per run needed to converge.

In Tables 3.10, 3.11 and 3.12 we report the results without preconditioner. All the test cases show Truncated AAR outperforming both Restarted GMRES(10) and Restarted GMRES(30) in terms of computational time, except for the matrix `chevron2` where the situation is flipped. In particular, Truncated AAR(6,10) is the only solver to converge on the matrices `fidapm37`, `fidap008`, `garon1`, `garon2` and `ABACUS_shell_hd`. Although the iteration count for AAR is usually higher than for the other linear solvers, the reader must remember that the computational cost for each iteration of AAR is not the same. In fact, most of the iterations are simply relaxation sweeps which are far cheaper than least-squares. The fact that Truncated AAR(6,10) takes less than Restarted GMRES(10) may be explained by the values of the angle $\angle(\mathbf{r}^{k+2m}, \mathbf{r}^k)$, which are higher for Truncated AAR(6,10) than for Restarted GMRES(10). Therefore, Truncated AAR(6,10) seems to favor linear independence between residual vectors after every other cycle. As concerns the comparison with Restarted GMRES(30), the situation is more complex. In fact, timings for Restarted GMRES(30) are higher than for Truncated AAR(6,10) although Restarted GMRES(30) produces a sequence of residual vectors that leads to higher values of $\angle(\mathbf{r}^{k+2m}, \mathbf{r}^k)$. A plausible explanation is that Truncated AAR(6,10) still outperforms Restarted GMRES(30) because of the simplicity of Richardson sweeps. In fact, the cheapness of the relaxation step may compensate for the need to run more iterations due to a lower value of $\angle(\mathbf{r}^{k+2m}, \mathbf{r}^k)$.

The situation is similar when a diagonal preconditioner is used. Results in Tables

	Truncated AAR($m = 6$, $p = 10$)				
Matrix	Time (s)	rel.error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.12	$7.89 \cdot 10^{-7}$	124	50.76	73.47
fidapm37	176.01	0.04	16,464	3.19	5.19
raefsky5	0.21	0.64	205	54.29	72.68
sp1	0.52	$2.05 \cdot 10^{-7}$	157	39.55	64.25
sp3	1.06	$2.08 \cdot 10^{-7}$	179	36.88	60.95
sp5	1.67	$1.91 \cdot 10^{-7}$	173	37.23	61.31
bcsstk29	-	-	-	-	-
sherman3	5.33	0.46	7,202	6.37	10.21
sherman5	4.37	$1.58 \cdot 10^{-5}$	8,057	7.71	11.06
fidap008	3.65	0.09	6,487	2.98	5.39
chipcool0	15.40	$3.42 \cdot 10^{-3}$	4,774	65.72	65.55
e20r0000	199.13	$1.25 \cdot 10^{-3}$	195,460	0.35	0.66
spmsrtls	-	-	-	-	-
III_Stokes	20.59	0.09	6,570	3.82	6.82
garon1	59.13	$2.71 \cdot 10^{-3}$	106,900	0.63	1.10
garon2	747.91	$7.44 \cdot 10^{-3}$	222,743	87.95	80.36
memplus	3.53	$9.84 \cdot 10^{-6}$	1,366	16.89	25.75
saylr4	1.25	$9.16 \cdot 10^{-3}$	2,025	88.53	89.79
xenon1	6.06	$4.66 \cdot 10^{-5}$	700	19.12	30.57
xenon2	23.55	$6.97 \cdot 10^{-5}$	734	18.65	29.78
venkat01	-	-	-	-	-
QC2534	-	-	-	-	-
mplate	-	-	-	-	-
waveguide3	93.10	$1.18 \cdot 10^{-5}$	10,587	4.85	8.07
ABACUS_shell_hd	703.22	0.68	91,130	0.38	0.69
light_in_tissue	4.01	$1.10 \cdot 10^{-5}$	379	24.85	39.88
kim1	92.75	$1.24 \cdot 10^{-5}$	5,476	70.19	70.85
chevron2	3,135.26	$1.57 \cdot 10^{-5}$	46,917	64.47	48.27

Table 3.10: Truncated AAR(6,10). Experiments without preconditioner.

	Restarted GMRES$(m = 10)$				
Matrix	Time (s)	rel. error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.20	$1.57 \cdot 10^{-6}$	137	71.44	30.12
fidapm37	-	-	-	-	-
raefsky5	0.27	0.64	96	75.77	61.42
sp1	1.09	$3.14 \cdot 10^{-7}$	200	62.80	24.70
sp3	1.67	$2.78 \cdot 10^{-7}$	217	61.14	22.48
sp5	2.64	$2.89 \cdot 10^{-7}$	231	59.56	20.87
bcsstk29	-	-	-	-	-
sherman3	-	-	-	-	-
sherman5	-	-	-	-	-
fidap008	-	-	-	-	-
chipcool0	-	-	-	-	-
e20r0000	-	-	-	-	-
spmsrtls	-	-	-	-	-
III_Stokes	-	-	-	-	-
garon1	-	-	-	-	-
garon2	-	-	-	-	-
memplus	48.81	$2.52 \cdot 10^{-5}$	13,555	7.34	0.51
saylr4	63.55	$8.81 \cdot 10^{-3}$	5,425	2.39	0.13
xenon1	76.95	$7.24 \cdot 10^{-5}$	7,378	9.94	1.27
xenon2	250.85	$9.43 \cdot 10^{-5}$	6,686	9.53	1.29
venkat01	-	-	-	-	-
QC2534	-	-	-	-	-
mplate	-	-	-	-	-
waveguide3	487.38	$6.67 \cdot 10^{-6}$	45,945	3.16	2.48
ABACUS_shell_hd	-	-	-	-	-
light_in_tissue	20.73	$1.64 \cdot 10^{-5}$	2,276	16.04	5.90
kim1	289.47	$1.26 \cdot 10^{-5}$	22,018	4.09	6.65
chevron2	1946.30	$1.14 \cdot 10^{-5}$	73,160	2.78	4.03

Table 3.11: Restarted GMRES(10). Experiments without preconditioner.

	Restarted GMRES$(m = 30)$				
Matrix	Time (s)	rel. error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.18	$1.15 \cdot 10^{-6}$	89	89.64	88.23
fidapm37	-	-	-	-	-
raefsky5	0.21	0.68	54	89.99	-
sp1	0.78	$2.21 \cdot 10^{-7}$	134	88.10	89.81
sp3	1.75	$2.07 \cdot 10^{-7}$	138	87.52	62.57
sp5	2.67	$2.07 \cdot 10^{-7}$	144	87.77	69.99
bcsstk29	-	-	-	-	-
sherman3	60.76	0.46	32,594	5.18	0.13
sherman5	22.45	$2.89 \cdot 10^{-5}$	21,862	9.33	5.26
fidap008	-	-	-	-	-
chipcool0	515.9	$3.33 \cdot 10^{-3}$	81,673	3.58	0.25
e20r0000	46.34	$7.39 \cdot 10^{-3}$	26,505	7.39	0.41
spmsrtls	-	-	-	-	-
III_Stokes	428.11	0.09	63,661	3.85	1.95
garon1	-	-	-	-	-
garon2	-	-	-	-	-
memplus	21.93	$2.51 \cdot 10^{-5}$	4,001	23.19	3.93
saylr4	20.17	$8.82 \cdot 10^{-3}$	15,017	7.96	1.23
xenon1	39.12	$7.15 \cdot 10^{-5}$	2,295	28.75	8.51
xenon2	137.89	$9.37 \cdot 10^{-5}$	2,394	27.29	8.15
venkat01	-	-	-	-	-
QC2534	-	-	-	-	-
mplate	-	-	-	-	-
waveguide3	221.74	$7.70 \cdot 10^{-6}$	18,654	9.56	10.95
ABACUS_shell_hd	-	-	-	-	-
light_in_tissue	14.09	$1.61 \cdot 10^{-5}$	829	40.16	19.40
kim1	246.59	$1.27 \cdot 10^{-5}$	12,828	9.14	14.72
chevron2	666.78	$1.35 \cdot 10^{-5}$	18,690	9.53	13.22

Table 3.12: Restarted GMRES(30). Experiments without preconditioner.

3.13, 3.14 and 3.15 show Truncated AAR(6,10) outperforming both Restarted GMRES(10) and Restarted GMRES(30). Moreover, AAR is the only method to converge (albeit very slowly) on the matrices `fidapm37`, `fidap008`, `garon2` and `venkat01`. As concerns Truncated AAR(6,10) vs. Restarted GMRES(10), the former does a better job in preserving linear independence as confirmed by the average values of the angles. On the other hand, Restarted GMRES(30), still takes longer to converge because of more expensive operations to perform in each cycle.

As concerns the use of ILU(0) as a preconditioner, results are reported in Tables 3.16, 3.17 and 3.18 only for matrices that allow a numerically stable construction of the preconditioner. The trend displayed in the tables is similar to the one already discussed in case of no preconditioner or diagonal preconditioner. In fact, Truncated AAR(6,10) outperforms Restarted GMRES(10) as well as Restarted GMRES(30). This is evident also in the performance profile in Figure 3.14. Through a comparison between Truncated AAR(6,10) and Restarted GMRES(10), it is possible to see that the former produces higher values of the average angles which explains the faster convergence. As concerns Restarted GMRES(30), the slower convergence than for Truncated AAR(6,10) is due to a more costly cycle between each restart.

The details about the performance of Truncated AAR(6,10), Restarted GMRES(10) and Restarted GMRES(30) when ILUT(10^{-4}) is used as a preconditioner are shown in Tables 3.19, 3.20 and 3.21. Also in this case, Truncated AAR(6,10) performs better than Restarted GMRES(10) and Restarted GMRES(30), as confirmed also by the performance profile in Figure 3.15. The discrepancies between the timings is not as accentuated as for the other preconditioning techniques, since the efficiency of ILUT(10^{-4}) is expected to cope with some deficiency of the solver itself. However, even for this highly efficient preconditioner, the reader can notice that there is still a test case where Truncated AAR(6,10) succeeds in converging, whereas Restarted GMRES(10) and Restarted GMRES(30) do not. This test case of interest is the

	Truncated AAR$(m = 6,\, p = 10)$				
Matrix	Time (s)	rel. error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.05	$8.07 \cdot 10^{-7}$	26	63.09	73.81
fidapm37	11.93	0.01	7,075	8.75	7.82
raefsky5	0.08	$7.19 \cdot 10^{-9}$	25	28.68	1.02
sp1	0.46	$2.39 \cdot 10^{-7}$	120	50.84	72.03
sp3	0.86	$3.16 \cdot 10^{-7}$	129	43.08	65.56
sp5	1.50	$3.04 \cdot 10^{-7}$	137	41.81	63.83
bcsstk29	-	-	-	-	-
sherman3	1.41	$1.12 \cdot 10^{-4}$	1,346	6.99	10.02
sherman5	0.24	$1.57 \cdot 10^{-6}$	261	35.26	51.72
fidap008	71.57	0.06	129,410	0.17	0.22
chipcool0	1.67	$6.21 \cdot 10^{-7}$	506	26.45	37.81
e20r0000	4.09	$2.70 \cdot 10^{-3}$	6,988	4.84	7.51
spmsrtls	-	-	-	-	-
III_Stokes	17.15	0.09	6,256	5.10	7.90
garon1	8.07	$1.05 \cdot 10^{-3}$	18,196	2.57	4.10
garon2	47.18	$4.31 \cdot 10^{-3}$	27,195	1.80	2.84
memplus	1.03	$3.66 \cdot 10^{-5}$	380	22.02	34.72
saylr4	6.21	$9.63 \cdot 10^{-3}$	1,684	13.56	20.11
xenon1	6.85	$2.61 \cdot 10^{-5}$	781	19.32	29.65
xenon2	26.88	$6.58 \cdot 10^{-5}$	818	20.32	30.63
venkat01	1,352.34	$2.43 \cdot 10^{-4}$	78,742	1.18	1.65
QC2534	-	-	-	-	-
mplate	-	-	-	-	-
waveguide3	138.09	$9.95 \cdot 10^{-6}$	14,134	4.40	6.98
ABACUS_shell_hd	-	-	-	-	-
light_in_tissue	5.08	$5.21 \cdot 10^{-6}$	361	31.19	45.08
kim1	2.49	$3.55 \cdot 10^{-7}$	145	47.07	63.46
chevron2	318.79	$9.14 \cdot 10^{-6}$	6,783	6.19	9.35

Table 3.13: Truncated AAR(6,10). Experiments with diagonal preconditioner.

	Restarted GMRES$(m = 10)$				
Matrix	Time (s)	rel.error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.05	$1.31 \cdot 10^{-8}$	31	15.04	9.04
fidapm37	-	-	-	-	-
raefsky5	0.07	$1.03 \cdot 10^{-8}$	19	2.64	-
sp1	0.71	$3.81 \cdot 10^{-7}$	132	68.48	37.14
sp3	1.14	$5.44 \cdot 10^{-7}$	154	62.44	30.88
sp5	2.14	$5.43 \cdot 10^{-7}$	171	60.58	39.33
bcsstk29	-	-	-	-	-
sherman3	20.39	$1.63 \cdot 10^{-4}$	4,684	4.76	0.88
sherman5	0.71	$2.82 \cdot 10^{-6}$	596	36.99	25.90
fidap008	-	-	-	-	-
chipcool0	6.07	$1.34 \cdot 10^{-5}$	1,407	44.99	11.62
e20r0000	-	-	-	-	-
spmsrtls	-	-	-	-	-
III_Stokes	-	-	-	-	-
garon1	-	-	-	-	-
garon2	-	-	-	-	-
memplus	3.17	$5.13 \cdot 10^{-5}$	808	12.82	5.52
saylr4	-	-	-	-	-
xenon1	35.53	$3.12 \cdot 10^{-5}$	3,033	14.83	7.80
xenon2	132.85	$9.39 \cdot 10^{-5}$	3,746	13.27	4.10
venkat01	-	-	-	-	-
QC2534	-	-	-	-	-
mplate	-	-	-	-	-
waveguide3	481.38	$6.72 \cdot 10^{-6}$	51,370	2.75	2.44
ABACUS_shell_hd	-	-	-	-	-
light_in_tissue	14.05	$2.05 \cdot 10^{-5}$	1,457	20.52	11.67
kim1	5.83	$3.07 \cdot 10^{-7}$	440	28.74	48.82
chevron2	247.19	$8.57 \cdot 10^{-6}$	11,059	7.24	10.27

Table 3.14: Restarted GMRES(10). Experiments with diagonal preconditioner.

	Restarted GMRES$(m=30)$				
Matrix	Time (s)	rel.error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.04	$7.08 \cdot 10^{-9}$	21	-	-
fidapm37	-	-	-	-	-
raefsky5	0.07	$6.6 \cdot 10^{-9}$	20	-	-
sp1	0.64	$2.95 \cdot 10^{-7}$	104	83	86.97
sp3	1.38	$3.89 \cdot 10^{-7}$	111	87.38	87.77
sp5	2.16	$4.16 \cdot 10^{-7}$	118	86.1	88
bcsstk29	-	-	-	-	-
sherman3	12.05	$1.75 \cdot 10^{-4}$	1,448	9.44	10.37
sherman5	0.73	$1.84 \cdot 10^{-7}$	123	68.47	44.18
fidap008	-	-	-	-	-
chipcool0	3.60	$5 \cdot 10^{-5}$	558	35.59	31.63
e20r0000	59.78	$3.45 \cdot 10^{-3}$	33,769	11.25	3.93
spmsrtls	-	-	-	-	-
III_Stokes	200.41	0.09	26,562	6.65	3.58
garon1	128.32	$1.43 \cdot 10^{-3}$	85,426	4.20	1.35
garon2	-	-	-	-	-
memplus	1.38	$5 \cdot 10^{-5}$	286	67.95	33.43
saylr4	15.36	$9.44 \cdot 10^{-3}$	11,328	10.85	4.75
xenon1	25.64	$3 \cdot 10^{-5}$	1,610	36.27	20.23
xenon2	111.51	$9.56 \cdot 10^{-5}$	1,872	29.17	10.23
venkat01	-	-	-	-	-
QC2534	-	-	-	-	-
mplate	-	-	-	-	-
waveguide3	254.93	$7.97 \cdot 10^{-6}$	19,731	9.44	10.87
ABACUS_shell_hd	-	-	-	-	-
light_in_tissue	13.58	$2.03 \cdot 10^{-5}$	1,006	40.63	17.19
kim1	6.81	$2.10 \cdot 10^{-7}$	390	68.23	72.88
chevron2	165.98	$9.35 \cdot 10^{-6}$	4,254	17.00	22.72

Table 3.15: Restarted GMRES(30). Experiments with diagonal preconditioner.

	Truncated AAR$(m = 6,\, p = 10)$				
Matrix	Time (s)	rel.error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.01	$4.6 \cdot 10^{-13}$	19	0.13	-
fidapm37*	-	-	-	-	-
raefsky5	0.07	$2.39 \cdot 10^{-11}$	7	88.03	-
sp1	0.29	$3.51 \cdot 10^{-8}$	53	66.94	82.05
sp3	0.57	$2.97 \cdot 10^{-8}$	56	67.31	87.38
sp5	0.72	$2.72 \cdot 10^{-8}$	43	65.59	85.72
bcsstk29*	$*$	$*$	$*$	$*$	$*$
sherman3	0.38	$4.28 \cdot 10^{-9}$	263	36.66	46.72
sherman5	0.23	$1.88 \cdot 10^{-7}$	61	59.34	78.51
fidap008*	-	-	-	-	-
chipcool0	0.52	$8.43 \cdot 10^{-8}$	117	40.19	44.44
e20r0000*	$*$	$*$	$*$	$*$	$*$
spmsrtls*	0.09	$1.69 \cdot 10^{-13}$	7	89.80	89.01
III_Stokes*	$*$	$*$	$*$	$*$	$*$
garon1*	$*$	$*$	$*$	$*$	$*$
garon1*	$*$	$*$	$*$	$*$	$*$
memplus	1.02	$1.22 \cdot 10^{-5}$	360	22.47	34.57
saylr4	0.15	$7.98 \cdot 10^{-9}$	123	19.82	21.22
xenon1*	-	-	-	-	-
xenon2*	-	-	-	-	-
venkat01	7.94	$7.98 \cdot 10^{-7}$	379	33.46	49.52
QC2534	350.57	$9.48 \cdot 10^{-3}$	90,685	-	-
mplate	-	-	-	-	-
waveguide3	253.67	$9.98 \cdot 10^{-7}$	18,211	4.76	6.98
ABACUS_shell_hd	-	-	-	-	-
light_in_tissue	1.73	$4.86 \cdot 10^{-7}$	114	53.48	73.18
kim1	0.33	$7.73 \cdot 10^{-9}$	14	46.91	89.40
chevron2	44.88	$9.30 \cdot 10^{-7}$	867	20.35	30.70

Table 3.16: Truncated AAR(6,10). Experiments with ILU(0) preconditioner.

	Restarted GMRES$(m = 10)$				
Matrix	Time (s)	rel. error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.02	$3.66 \cdot 10^{-9}$	6	-	-
fidapm37*	-	-	-	-	-
raefsky5	0.03	$3.98 \cdot 10^{-11}$	4	-	-
sp1	0.17	$4.36 \cdot 10^{-7}$	40	78.45	78.40
sp3	1.08	$4.4 \cdot 10^{-8}$	41	82.91	88.47
sp5	2.12	$5.38 \cdot 10^{-8}$	39	82.67	88.07
sherman3	1.24	$7.42 \cdot 10^{-6}$	644	25.22	12.28
sherman5	0.14	$4.36 \cdot 10^{-7}$	86	73.66	57.57
fidap008*	-	-	-	-	-
chipcool0	1.54	$2.55 \cdot 10^{-7}$	190	75.51	39.68
e20r0000*	$*$	$*$	$*$	$*$	$*$
spmsrtls*	0.03	$5.99 \cdot 10^{-14}$	1	$*$	$*$
III_Stokes*	$*$	$*$	$*$	$*$	$*$
garon1*	$*$	$*$	$*$	$*$	$*$
garon2*	$*$	$*$	$*$	$*$	$*$
memplus	4.63	$1.37 \cdot 10^{-5}$	813	19.35	6.79
saylr4	0.54	$1.22 \cdot 10^{-8}$	317	46.69	20.25
xenon1*	-	-	-	-	-
xenon2*	-	-	-	-	-
venkat01	11.85	$7.75 \cdot 10^{-7}$	336	47.38	65.30
QC2534*	-	-	-	-	-
mplate	-	-	-	-	-
waveguide3	-	-	-	-	-
ABACUS_shell_hd	-	-	-	-	-
light_in_tissue	4.80	$1.89 \cdot 10^{-6}$	264	49.14	24.72
kim1	0.49	$5.69 \cdot 10^{-9}$	13	88.03	-
chevron2	59.70	$7.35 \cdot 10^{-7}$	1,179	16.75	23.38

Table 3.17: Restarted GMRES(10). Experiments with ILU(0) preconditioner.

	Restarted GMRES$(m=30)$				
Matrix	Time (s)	rel. error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.02	$2.42 \cdot 10^{-9}$	6	-	-
fidapm37*	-	-	-	-	-
raefsky5	0.03	$3.98 \cdot 10^{-11}$	4	-	-
sp1	0.62	$4.46 \cdot 10^{-8}$	39	89.03	-
sp3	1.27	$5.67 \cdot 10^{-8}$	36	87.93	-
sp5	2.12	$4.97 \cdot 10^{-8}$	35	89.30	-
bcsstk29*	*	*	*	*	*
sherman3	0.58	$4.15 \cdot 10^{-6}$	228	52.63	39.70
sherman5	0.12	$1.94 \cdot 10^{-7}$	33	83.47	-
fidap008*	-	-	-	-	-
chipcool0	0.92	$8.1 \cdot 10^{-7}$	79	86.40	89.74
e20r0000*	*	*	*	*	*
spmsrtls*	0.03	$5.99 \cdot 10^{-14}$	1	-	-
III_Stokes	*	*	*	*	*
garon1*	*	*	*	*	*
garon2*	*	*	*	*	*
memplus	2.69	$1.69 \cdot 10^{-5}$	344	47.10	26.96
saylr4	0.24	$2.53 \cdot 10^{-6}$	135	3.59	3.46
xenon1*	-	-	-	-	-
xenon2*	-	-	-	-	-
venkat01	13.94	$7.67 \cdot 10^{-7}$	318	71.37	75.54
QC2534*	-	-	-	-	-
mplate	-	-	-	-	-
waveguide3	-	-	-	-	-
ABACUS_shell_hd	-	-	-	-	-
light_in_tissue	3.72	$1-35 \cdot 10^{-6}$	192	82.72	52.15
kim1	0.49	$5.69 \cdot 10^{-9}$	13	-	-
chevron2	171.75	$8.75 \cdot 10^{-7}$	2.610	13.08	23.21

Table 3.18: Restarted GMRES(30). Experiments with ILU(0) preconditioner.

matrix ABACUS_shell_hd.

3.3.8 Comparison between Truncated AAR and Augmented AAR

The analysis of Truncated AAR(6,10), Restarted GMRES(10) and Restarted GMRES(30) has revealed empirical evidence that Truncated AAR can be competitive against Restarted GMRES on a broad class of problems and for different preconditioners as well. In this section we aim to compare Truncated AAR with Augmented AAR, setting $m = 6$ and using two different values of p. In particular, we consider the cases when $p = 10$ and $p = 12$. The results are presented again as average over 10 runs. The solution is generated as a random vector for each run and the right hand side is consequently obtained multiplying the solution vector by the coefficient matrix A. The convergence check is still based on the relative residual ℓ^2-norm smaller than 10^{-8}. Experiments are run without preconditioner, with a diagonal preconditioner, with ILU(0) and with ILUT(10^{-4}). Results for ILU(0) and ILUT(10^{-4}) as a preconditioner are shown if the preconditioner can be constructed in a numerically stable way. For ease of exposition, we will refer to the variants of the algorithms used here as Truncated AAR(6,10), Truncated AAR(6,12), Augmented AAR(6,10) and Augmented AAR(6,12).

Results for Truncated AAR(6,12) are displayed in Tables 3.22, 3.23 and 3.24. A comparison between the results for Truncated AAR(6,12) and Truncated AAR(6,10) reveals similar performances for most of the test cases.

Results for Augmented AAR(6,10) are shown in Tables 3.26, 3.27, 3.28 and 3.29. The performance of Augmented AAR(6,10) is generally similar to the one of Truncated AAR(6,12). However, a significant deterioration of the performance is identified for the matrix sherman3 without preconditioner. Indeed Augmented AAR(6,10) takes almost one thousand times as Truncated AAR(6,10) and Truncated AAR(6,12)

	Truncated AAR($m = 6$, $p = 10$)				
Matrix	Time (s)	rel.error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.06	$7.48 \cdot 10^{-12}$	7	87.34	89.31
fidapm37*	$*$	$*$	$*$	$*$	$*$
rafesky5	0.04	$1.92 \cdot 10^{-11}$	7	78.12	85.13
sp1	0.13	$1.20 \cdot 10^{-9}$	7	83.65	88.54
sp3	0.16	$2.14 \cdot 10^{-9}$	7	86.42	85.35
sp5	0.22	$6.85 \cdot 10^{-9}$	8	81.85	87.60
bcsstk29*	$*$	$*$	$*$	$*$	$*$
sherman3	0.10	$2.52 \cdot 10^{-10}$	13	3.17	89.56
sherman5	0.08	$1.84 \cdot 10^{-9}$	13	2.58	89.96
fidap008*	-	-	-	-	-
chipcool0	0.29	$6.72 \cdot 10^{-9}$	16	47.35	89.79
e20r0000	0.16	$1.11 \cdot 10^{-5}$	27	58.14	37.68
spmsrtls	0.07	$5.24 \cdot 10^{-11}$	7	4.56	89.55
III_Stokes	0.60	$8.34 \cdot 10^{-5}$	36	4.69	1.82
garon1	0.11	$7.24 \cdot 10^{-8}$	22	48.45	22.95
garon2	0.82	$1.36 \cdot 10^{-7}$	37	35.95	38.54
memplus	0.09	$4.05 \cdot 10^{-10}$	13	3.65	89.87
saylr4	0.13	$1.35 \cdot 10^{-6}$	107	48.44	67.22
xenon1	0.91	$9.16 \cdot 10^{-10}$	25	63.22	89.12
xenon2	2.99	$1.95 \cdot 10^{-8}$	27	66.85	89.23
venkat01	1.94	$3.17 \cdot 10^{-10}$	19	82.56	88.58
QC2534	0.14	$1.11 \cdot 10^{-11}$	13	4.32	89.93
mplate	16.41	0.04	2,048	12.24	18.02
waveguide3	0.76	$6.16 \cdot 10^{-9}$	13	57.60	82.78
ABACUS_shell_hd	1.14	9.05	72	37.58	30.41
light_in_tissue	0.34	$7.31 \cdot 10^{-9}$	15	45.19	89.99
kim1	0.23	$3.87 \cdot 10^{-9}$	7	56.12	87.23
chevron2	4.56	$3.19 \cdot 10^{-8}$	61	58.29	74.70

Table 3.19: Truncated AAR(6,10). Experiments with ILUT(10^{-4}) preconditioner.

	Restarted GMRES$(m = 10)$				
Matrix	Time (s)	rel.error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.01	$1.27 \cdot 10^{-11}$	3	-	-
fidapm37*	-	-	-	-	-
raefsky5	0.03	$1.34 \cdot 10^{-10}$	3	-	-
sp1	0.08	$6.20 \cdot 10^{-9}$	4	-	-
sp3	0.20	$2.09 \cdot 10^{-10}$	5	-	-
sp5	0.29	$1.99 \cdot 10^{-10}$	5	-	-
bcsstk29*	$*$	$*$	$*$	$*$	$*$
sherman3	0.04	$2.03 \cdot 10^{-9}$	8	-	-
sherman5	0.02	$2.31 \cdot 10^{-9}$	8	-	-
fidap008*	-	-	-	-	-
chipcool0	0.48	$7.56 \cdot 10^{-9}$	12	89.99	-
e20r0000	0.22	$8.39 \cdot 10^{-5}$	26	89.94	87.74
spmsrtls	0.05	$1.05 \cdot 10^{-10}$	3	-	-
III_Stokes	1.32	0.06	30	88.72	89.99
garon1	0.15	$2.70 \cdot 10^{-8}$	17	89.99	-
garon2	1.15	$2.13 \cdot 10^{-7}$	34	89.45	89.53
memplus	0.06	$2.86 \cdot 10^{-9}$	8	-	-
saylr4	0.53	$3.09 \cdot 10^{-6}$	289	42.52	17.95
xenon1	1.40	$1.26 \cdot 10^{-8}$	18	89.99	-
xenon2	5.38	$1.90 \cdot 10^{-8}$	25	89.98	-
venkat01	2.63	$5.51 \cdot 10^{-8}$	12	89.99	-
QC2534	0.07	$4.10 \cdot 10^{-9}$	10	-	-
mplate	191.99	0.05	13,935	7.32	8.39
waveguide3	1.30	$6.09 \cdot 10^{-9}$	14	83.83	-
ABACUS_shell_hd	-	-	-	-	-
light_in_tissue	0.39	$1.14 \cdot 10^{-8}$	13	88.78	-
kim1	0.33	$9.70 \cdot 10^{-10}$	8	-	-
chevron2	14.74	$2.87 \cdot 10^{-8}$	139	73.69	64.99

Table 3.20: Restarted GMRES(10). Experiments with ILUT(10^{-4}) preconditioner.

	Restarted GMRES$(m = 30)$				
Matrix	Time (s)	rel.error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.01	$1.27 \cdot 10^{-11}$	3	-	-
fidapm37*	-	-	-	-	-
raefsky5	0.03	$1.34 \cdot 10^{-10}$	3	-	-
sp1	0.08	$6.20 \cdot 10^{-9}$	4	-	-
sp3	0.20	$2.09 \cdot 10^{-10}$	5	-	-
sp5	0.29	$1.99 \cdot 10^{-10}$	5	-	-
bcsstk29*	$*$	$*$	$*$	$*$	$*$
sherman3	0.04	$2.03 \cdot 10^{-9}$	8	-	-
sherman5	0.02	$2.31 \cdot 10^{-9}$	8	-	-
fidap008*	-	-	-	-	-
chipcool0	0.44	$2.95 \cdot 10^{-9}$	11	-	-
e20r0000	0.13	$4.82 \cdot 10^{-5}$	16	-	-
spmsrtls	0.05	$1.05 \cdot 10^{-10}$	3	-	-
III_Stokes	0.84	$3.94 \cdot 10^{-4}$	15	-	-
garon1	0.12	$2.74 \cdot 10^{-8}$	15	-	-
garon2	1.00	$1.61 \cdot 10^{-7}$	26	-	-
memplus	0.06	$2.86 \cdot 10^{-9}$	8	-	-
saylr4	0.15	$1.21 \cdot 10^{-7}$	47	89.99	-
xenon1	1.27	$7.33 \cdot 10^{-9}$	17	-	-
xenon2	5.13	$1.44 \cdot 10^{-8}$	20	-	-
venkat01	2.36	$4.48 \cdot 10^{-8}$	12	-	-
QC2534	0.07	$4.10 \cdot 10^{-9}$	10	-	-
mplate	33.51	0.05	2,574	24.80	25.64
waveguide3	1.15	$6.72 \cdot 10^{-9}$	14	-	-
ABACUS_shell_hd	-	-	-	-	-
light_in_tissue	0.37	$6.08 \cdot 10^{-9}$	13	-	-
kim1	0.33	$9.70 \cdot 10^{-10}$	8	-	-
chevron2	6.33	$3.17 \cdot 10^{-8}$	57	87.72	89.72

Table 3.21: Restarted GMRES(30). Experiments with ILUT(10^{-4}) preconditioner.

	Truncated AAR($m = 6$, $p = 12$)				
Matrix	Time (s)	rel.error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.08	$8.78 \cdot 10^{-7}$	114	52.36	74.94
fidapm37	59.55	0.04	16,175	3.58	5.61
raefsky5	0.18	0.67	127	54.35	72.77
sp1	0.78	$2.11 \cdot 10^{-7}$	135	42.34	65.56
sp3	1.91	$1.68 \cdot 10^{-7}$	140	40.81	64.08
sp5	3.10	$1.85 \cdot 10^{-7}$	144	40.81	64.08
bcsstk29	-	-	-	-	-
sherman3	3.44	0.46	5,397	6.31	10.28
sherman5	10.48	$2.13 \cdot 10^{-5}$	17,630	4.89	7.04
fidap008	3.88	0.09	6,549	4.21	6.91
chipcool0	27.15	$3.07 \cdot 10^{-3}$	4,698	65.01	65.44
e20r0000	24.1	$7,45 \cdot 10^{-3}$	2,024	8.24	13.58
spmsrtls	-	-	-	-	-
III_Stokes	21.13	0.09	7,263	4.11	6.90
garon1	3.30	$6.50 \cdot 10^{-3}$	6,378	4.66	7.80
garon2	35.73	$8.27 \cdot 10^{-3}$	10,235	2.94	4.89
memplus	4.45	$1.80 \cdot 10^{-5}$	1,108	18.35	27.79
saylr4	1.03	$9.15 \cdot 10^{-3}$	1,853	88.61	89.94
xenon1	13.49	$3.77 \cdot 10^{-5}$	703	19.54	31.03
xenon2	25.59	$6.08 \cdot 10^{-5}$	737	19.19	30.45
venkat01	-	-	-	-	-
QC2534	-	-	-	-	-
mplate	-	-	-	-	-
waveguide3	93.70	$1.33 \cdot 10^{-5}$	9,189	4.85	8.07
ABACUS_shell_hd	878.56	0.07	86,710	-	-
light_in_tissue	4.19	$8.64 \cdot 10^{-6}$	342	25.88	41.43
kim1	132.09	$1.24 \cdot 10^{-5}$	4,928	76.10	77.54
chevron2	1,034.22	$2.00 \cdot 10^{-5}$	20,959	67.66	51.61

Table 3.22: Truncated AAR(6,12). Experiments without preconditioner.

	Truncated AAR($m=6$, $p=12$)				
Matrix	Time (s)	rel.error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.07	$3.91 \cdot 10^{-9}$	25	73.58	89.51
fidapm37	15.42	0.01	5,992	15.84	15.57
raefsky5	0.07	$2.38 \cdot 10^{-9}$	25	81.15	81.59
sp1	0.62	$2.99 \cdot 10^{-7}$	119	55.52	75.13
sp3	1.14	$3.67 \cdot 10^{-7}$	127	53.21	73.22
sp5	1.88	$3.02 \cdot 10^{-7}$	134	50.94	70.88
bcsstk29	-	-	-	-	-
sherman3	2.04	$1.47 \cdot 10^{-4}$	1,819	3.68	5.67
sherman5	0.21	$1.63 \cdot 10^{-6}$	245	33.64	51.64
fidap008	60.60	0.09	120,760	0.58	0.95
chipcool0	1.64	$1.13 \cdot 10^{-5}$	386	0.40	0.65
e20r0000	3.75	$3.16 \cdot 10^{-3}$	5,704	2.98	4.56
spmsrtls	-	-	-	-	-
III_Stokes	17,76	0.08	6,551	4.66	7.14
garon1	7.04	$1.16 \cdot 10^{-3}$	14,495	2.63	4.13
garon2	42.81	$4.41 \cdot 10^{-3}$	19,539	2.26	3.51
memplus	1.31	$4.51 \cdot 10^{-5}$	333	34.34	46.75
saylr4	1.18	$9.71 \cdot 10^{-3}$	1,878	13.55	20.32
xenon1	9.15	$4.40 \cdot 10^{-5}$	782	20.69	31.22
xenon2	32.86	$6.45 \cdot 10^{-5}$	822	20.34	30.66
venkat01	-	-	-	-	-
QC2534	-	-	-	-	-
mplate	-	-	-	-	-
waveguide3	178.02	$1.68 \cdot 10^{-5}$	13,148	3.88	6.16
ABACUS_shell_hd	-	-	-	-	-
light_in_tissue	6.87	$5.04 \cdot 10^{-6}$	360	31.17	45.04
kim1	3.20	$2.30 \cdot 10^{-7}$	139	48.79	67.16
chevron2	257.63	$1.23 \cdot 10^{-5}$	4,164	8.39	12.72

Table 3.23: Truncated AAR(6,12). Experiments with diagonal preconditioner.

	Truncated AAR$(m = 6,\, p = 12)$				
Matrix	Time (s)	rel.error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.04	$1.80 \cdot 10^{-12}$	13	1.99	89.99
fidapm37*	-	-	-	-	-
raefsky5	0.07	$2.05 \cdot 10^{-10}$	7	87.13	-
sp1	0.28	$1.08 \cdot 10^{-8}$	43	70.41	89.44
sp3	0.48	$4.48 \cdot 10^{-8}$	44	68.10	88.59
sp5	0.76	$3.67 \cdot 10^{-8}$	43	69.36	88.20
bcsstk29*	$*$	$*$	$*$	$*$	$*$
sherman3	0.26	$4.27 \cdot 10^{-6}$	206	41.99	61.19
sherman5	0.10	$2.86 \cdot 10^{-7}$	52	64.36	85.49
fidap008*	-	-	-	-	-
chipcool0	0.58	$6.80 \cdot 10^{-7}$	102	56.61	76.39
e20r0000*	$*$	$*$	$*$	$*$	$*$
spmsrtls*	0.11	$6.54 \cdot 10^{-14}$	7	23.19	86.54
III_Stokes*	$*$	$*$	$*$	$*$	$*$
garon1*	$*$	$*$	$*$	$*$	$*$
garon2*	$*$	$*$	$*$	$*$	$*$
memplus	1.19	$1.08 \cdot 10^{-5}$	330	33.45	47.70
saylr4	0.13	$1.74 \cdot 10^{-6}$	116	54.15	76.42
xenon1*	-	-	-	-	-
xenon2*	-	-	-	-	-
venkat01	11.14	$7.50 \cdot 10^{-7}$	364	33.86	49.28
QC2534	332.67	0.01	88,672	1.30	2.21
mplate	-	-	-	-	-
waveguide3	181.35	$1.81 \cdot 10^{-6}$	13,551	5.56	7.94
ABACUS_shell_hd	-	-	-	-	-
light_in_tissue	2.54	$3.77 \cdot 10^{-7}$	111	53.53	73.23
kim1	0.37	$7.73 \cdot 10^{-9}$	14	46.91	89.40
chevron2	44.91	$9.20 \cdot 10^{-7}$	839	20.91	31.46

Table 3.24: Truncated AAR(6,12). Experiments with ILU(0) preconditioner.

	Truncated AAR$(m = 6,\, p = 12)$				
Matrix	Time (s)	rel.error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.07	$7.48 \cdot 10^{-12}$	7	87.21	89.71
fidapm37*	-	-	-	-	-
raefsky5	0.07	$2.05 \cdot 10^{-10}$	7	82.34	89.10
sp1	0.15	$1.20 \cdot 10^{-9}$	7	80.32	89.43
sp3	0.17	$2.14 \cdot 10^{-9}$	7	84.11	89.56
sp5	0.40	$6.85 \cdot 10^{-9}$	8	81.10	89.73
bcsstk29*	$*$	$*$	$*$	$*$	$*$
sherman3	0.11	$2.55 \cdot 10^{-10}$	13	3.17	89.56
sherman5	0.08	$1.84 \cdot 10^{-9}$	13	2.58	89.96
fidap008*	-	-	-	-	-
chipcool0	0.32	$6.71 \cdot 10^{-9}$	16	47.35	89.79
e20r0000	0.17	$4.98 \cdot 10^{-6}$	26	58.14	33.97
spmsrtls	0.10	$5.23 \cdot 10^{-11}$	7	83.21	89.45
III_Stokes	0.64	$6.22 \cdot 10^{-5}$	29	6.94	3.48
garon1	0.13	$1.65 \cdot 10^{-8}$	19	48.45	22.86
garon2	0.70	$6.81 \cdot 10^{-8}$	37	42.61	30.86
memplus	0.12	$1.31 \cdot 10^{-10}$	13	3.57	89.16
saylr4	0.17	$1.09 \cdot 10^{-6}$	82	56.40	78.29
xenon1	1.21	$8.31 \cdot 10^{-10}$	25	61.45	89.92
xenon2	4.01	$5.91 \cdot 10^{-9}$	30	65.60	89.53
venkat01	2.04	$6.90 \cdot 10^{-10}$	19	82.56	88.58
QC2534	0.16	$1.11 \cdot 10^{-11}$	13	4.32	89.93
mplate	81.63	0.03	99,484	8.22	13.09
waveguide3	0.80	$6.16 \cdot 10^{-9}$	13	57.60	82.78
ABACUS_shell_hd	1.15	13.55	53	42.84	27.68
light_in_tissue	0.43	$7.31 \cdot 10^{-9}$	15	45.19	89.99
kim1	0.22	$3.87 \cdot 10^{-9}$	7	78.21	89.59
chevron2	4.44	$3.24 \cdot 10^{-8}$	56	52.80	65.90

Table 3.25: Truncated AAR(6,12). Experiments with ILU(10^{-4}) preconditioner.

to converge. The worsening of the performance is validated by the averaged values of the angles $\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$ and $\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$ that are less than 1. The performance is significantly worsened also for the matrix `garon2`, since the timing increases by a factor of 10 with respect to Truncated AAR(6,10) and Truncated AAR(6,12).

The performance of Augmented AAR(6,12) is shown in Tables 3.30, 3.31, 3.32 and 3.33. In general Augmented AAR(6,12) outperforms any other version of AAR employed in this section. Differently from Augmented AAR(6,10), the performance of Augmented AAR(6,12) is significantly improved for the matrix `sherman3` without preconditioner. In fact the time recorded for Augmented AAR(6,12) is comparable with Truncated AAR(6,10) and Truncated AAR(6,12).

To efficiently summarize the comparison between the performance of Trunctaed AAR(6,10), Trunctaed AAR(6,12), Augmented AAR(6,10) and Augmented AAR(6,12) we show the performance profiles in Figures 3.16, 3.17, 3.18 and 3.19 for no use of preconditioner, diagonal preconditioner, ILU(0) and ILUT(10^{-4}). Based on the evidence displayed in this graphs, we propose Augmented AAR(6,12) as the variant of Alternating Anderson-Richardson to adopt. The motivations behind this choice are essentially two. Firstly, Augmented AAR(6,12) usually outperforms any other version of AAR tested. Secondly, it allows to reduce the number of parameters that the user has to tune, since in this case $p = 2m$. Therefore, this version of the algorithm is more practical to use.

3.4 Conclusions and future work

In this chapter we have explored different techniques that can be used to accelerate one-level standard relaxation algorithms. The techniques that have been broadly studied in the literature are aggregation-disaggregation (including multigrid) algorithms and Anderson mixing. Although Anderson mixing is widely recognized as

	Augmented AAR$(m = 6,\, p = 10)$				
Matrix	Time (s)	rel.error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.10	$1.01 \cdot 10^{-6}$	115	51.79	74.83
fidapm37	33.36	0.04	16,949	3.68	5.78
raefsky5	0.18	0.67	135	52.72	70.46
sp1	0.50	$1.84 \cdot 10^{-7}$	144	41.85	66.73
sp3	1.24	$2.09 \cdot 10^{-7}$	158	41.09	65.06
sp5	1.49	$1.80 \cdot 10^{-7}$	151	41.42	65.55
bcsstk29	-	-	-	-	-
sherman3	176.72	0.46	255,484	0.57	0.59
sherman5	4.22	$1.53 \cdot 10^{-5}$	10,233	5.88	8.40
fidap008	3.70	0.08	8,677	3.46	6.13
chipcool0	14.54	$3.33 \cdot 10^{-3}$	5,214	71.53	68.23
e20r0000	6.05	$6.74 \cdot 10^{-3}$	10,090	3.77	6.29
spmsrtls	-	-	-	-	-
III_Stokes	21.05	0.09	6,666	4.34	7.39
garon1	24.02	$5.17 \cdot 10^{-3}$	58,338	0.60	1.05
garon2	359.34	$8.30 \cdot 10^{-3}$	148,348	0.36	0.61
memplus	2.55	$1.25 \cdot 10^{-5}$	1,227	17.13	25.91
saylr4	1.16	$9.10 \cdot 10^{-3}$	2,531	87.88	81.89
xenon1	6.41	$3.68 \cdot 10^{-5}$	702	20.64	32.93
xenon2	33.39	$6.94 \cdot 10^{-5}$	723	21.43	33.22
venkat01	-	-	-	-	-
QC2534	-	-	-	-	-
mplate	-	-	-	-	-
waveguide3	105.33	$1.19 \cdot 10^{-5}$	10,200	4.44	7.38
ABACUS_shell_hd	-	-	-	-	-
light_in_tissue	3.94	$1.13 \cdot 10^{-5}$	360	26.68	43.77
kim1	102.09	$1.26 \cdot 10^{-5}$	55,546	73.38	71.44
chevron2	2,530.65	$1.57 \cdot 10^{-5}$	46,352	75.96	60.83

Table 3.26: Augmented AAR(6,10). Experiments without preconditioner.

	Augmented AAR$(m = 6,\, p = 10)$				
Matrix	Time (s)	rel.error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.06	$4.00 \cdot 10^{-9}$	25	69.69	89.42
fidapm37	15.63	$9.97 \cdot 10^{-3}$	6,240	15.35	18.98
raefsky5	0.10	$4.01 \cdot 10^{-9}$	25	77.69	87.86
sp1	0.71	$2.28 \cdot 10^{-7}$	122	51.71	73.73
sp3	0.90	$3.20 \cdot 10^{-7}$	128	51.33	72.10
sp5	1.84	$3.06 \cdot 10^{-7}$	135	49.63	70.61
bcsstk29	-	-	-	-	-
sherman3	1.37	$1.47 \cdot 10^{-4}$	1,434	14.34	21.93
sherman5	0.22	$1.57 \cdot 10^{-6}$	266	32.01	50.76
fidap008	136.18	0.08	271,222	0.41	0.68
chipcool0	1.98	$8.24 \cdot 10^{-6}$	491	25.63	38.35
e20r0000	4.45	$2.56 \cdot 10^{-3}$	7,390	3.03	4.67
spmsrtls	-	-	-	-	-
III_Stokes	21.28	0.09	6,069	4.83	7.59
garon1	9.71	$1.12 \cdot 10^{-3}$	22,741	1.70	2.70
garon2	61.23	$4.42 \cdot 10^{-3}$	34,065	1.17	2.80
memplus	1.33	$5.27 \cdot 10^{-5}$	369	30.87	43.29
saylr4	0.96	$9.70 \cdot 10^{-3}$	1,615	13.34	20.04
xenon1	8.20	$4.37 \cdot 10^{-5}$	777	21.22	32.19
xenon2	28.90	$6.54 \cdot 10^{-5}$	818	20.68	31.44
venkat01	1,764.26	$4.34 \cdot 10^{-4}$	93,814	1.33	1.84
QC2534	-	-	-	-	-
mplate	-	-	-	-	-
waveguide3	128.03	$1.00 \cdot 10^{-5}$	14,372	4.25	6.77
ABACUS_shell_hd	-	-	-	-	-
light_in_tissue	5.88	$5.01 \cdot 10^{-6}$	361	31.22	45.44
kim1	2.89	$2.86 \cdot 10^{-7}$	145	50.46	67.16
chevron2	316.31	$9.53 \cdot 10^{-6}$	6,692	6.39	9.70

Table 3.27: Augmented AAR(6,10). Experiments with diagonal preconditioner.

	Augmented AAR$(m = 6,\, p = 10)$				
Matrix	Time (s)	rel.error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.08	$1.81 \cdot 10^{-12}$	13	1.99	89.99
fidapm37*	-	-	-	-	-
raefsky5	0.04	$3.39 \cdot 10^{-10}$	7	88.01	-
sp1	0.26	$1.38 \cdot 10^{-8}$	43	70.53	89.16
sp3	0.49	$4.03 \cdot 10^{-8}$	43	68.95	89.19
sp5	0.71	$3.00 \cdot 10^{-8}$	43	68.55	87.91
bcsstk29*	$*$	$*$	$*$	$*$	$*$
sherman3	0.40	$4.17 \cdot 10^{-6}$	250	41.30	60.31
sherman5	0.07	$9.83 \cdot 10^{-8}$	63	59.81	85.54
fidap008*	-	-	-	-	-
chipcool0	0.61	$9.33 \cdot 10^{-7}$	127	46.96	68.54
e20r0000*	$*$	$*$	$*$	$*$	$*$
spmsrtls*	0.07	$9.48 \cdot 10^{-14}$	7	89.10	-
III_Stokes*	$*$	$*$	$*$	$*$	$*$
garon1*	$*$	$*$	$*$	$*$	$*$
garon2*	$*$	$*$	$*$	$*$	$*$
memplus	1.28	$1.19 \cdot 10^{-5}$	357	31.89	46.92
saylr4	0.13	$2.35 \cdot 10^{-6}$	125	45.82	68.09
xenon1*	-	-	-	-	-
xenon2*	-	-	-	-	-
venkat01	6.31	$7.76 \cdot 10^{-7}$	379	34.33	50.54
QC2534	329.21	0.03	91,283	0.38	0.65
mplate	-	-	-	-	-
waveguide3	271.71	$1.03 \cdot 10^{-6}$	17,011	5.35	7.85
ABACUS_shell_hd	-	-	-	-	-
light_in_tissue	1.71	$5.40 \cdot 10^{-7}$	113	53.14	73.80
kim1	0.34	$7.18 \cdot 10^{-9}$	14	46.85	89.31
chevron2	37.25	$8.90 \cdot 10^{-7}$	867	20.33	30.77

Table 3.28: Augmented AAR(6,10). Experiments with ILU(0) preconditioner.

	Augmented AAR$(m = 6,\, p = 10)$				
Matrix	Time (s)	rel.error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.07	$7.48 \cdot 10^{-12}$	7	84.12	89.21
fidapm37*	-	-	-	-	-
raefsky5	0.04	$1.40 \cdot 10^{-12}$	7	88.01	-
sp1	0.14	$1.20 \cdot 10^{-9}$	7	88.34	88.43
sp3	0.15	$2.14 \cdot 10^{-9}$	7	85.76	89.99
sp5	0.20	$6.85 \cdot 10^{-9}$	8	81.13	89.01
bcsstk29*	$*$	$*$	$*$	$*$	$*$
sherman3	0.09	$2.38 \cdot 10^{-10}$	13	2.39	89.99
sherman5	0.08	$2.23 \cdot 10^{-9}$	13	2.58	89.99
fidap008*	-	-	-	-	-
chipcool0	0.37	$6.26 \cdot 10^{-9}$	16	47.34	89.75
e20r0000*	0.17	$3.42 \cdot 10^{-5}$	73	59.51	27.89
spmsrtls*	0.10	$5.24 \cdot 10^{-11}$	7	84.55	89.13
III_Stokes	0.75	$1.55 \cdot 10^{-5}$	43	2.61	2.02
garon1	0.13	$6.34 \cdot 10^{-8}$	20	32.78	28.47
garon2	0.75	$2.14 \cdot 10^{-8}$	43	44.10	45.54
memplus	0.11	$1.30 \cdot 10^{-10}$	13	3.65	89.86
saylr4	0.17	$1.78 \cdot 10^{-6}$	107	51.50	67.46
xenon1	1.00	$1.14 \cdot 10^{-9}$	25	60.71	83.74
xenon2	4.03	$7.43 \cdot 10^{-9}$	31	66.52	89.36
venkat01	2.00	$3.68 \cdot 10^{-10}$	19	82.73	88.48
QC2534	0.13	$2.27 \cdot 10^{-10}$	13	4.32	89.92
mplate	15.21	0.04	2,031	13.28	21.80
waveguide3	0.79	$5.85 \cdot 10^{-9}$	13	29.67	82.11
ABACUS_shell_hd	1.20	17.35	65	38.34	32.43
light_in_tissue	0.32	$7.34 \cdot 10^{-9}$	15	45.19	89.99
kim1	0.19	$3.84 \cdot 10^{-9}$	7	89.33	89.92
chevron2	4.41	$2.93 \cdot 10^{-8}$	60	56.14	57.56

Table 3.29: Augmented AAR(6,10). Experiments with ILUT(10^{-4}) preconditioner.

	Augmented AAR$(m = 6,\ p = 12)$				
Matrix	Time (s)	rel.error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.06	$9.34 \cdot 10^{-7}$	114	53.02	75.39
fidapm37	58.69	0.04	16,902	3.63	5.73
raefsky5	0.15	0.67	129	54.22	72.20
sp1	0.78	$2.19 \cdot 10^{-7}$	140	42.08	67.32
sp3	1.95	$1.68 \cdot 10^{-7}$	140	40.75	65.58
sp5	3.16	$2.10 \cdot 10^{-7}$	147	41.06	66.09
bcsstk29	-	-	-	-	-
sherman3	3.43	0.46	5,522	6.43	10.69
sherman5	4.41	$1.58 \cdot 10^{-5}$	10,730	6.59	9.53
fidap008	3.25	0.09	6,212	4.41	7.13
chipcool0	12.30	$3.21 \cdot 10^{-3}$	4,570	79.57	79.12
e20r0000	0.93	$7.36 \cdot 10^{-3}$	1,237	7.96	12.95
spmsrtls	-	-	-	-	-
III_Stokes	23.35	0.09	6,804	4.39	7.24
garon1	6.03	$5.76 \cdot 10^{-3}$	12,287	4.51	7.51
garon2	30.93	$8.61 \cdot 10^{-3}$	12,616	1.94	3.20
memplus	3.30	$1.38 \cdot 10^{-5}$	1,148	17.82	27.44
saylr4	0.99	$9.14 \cdot 10^{-3}$	1,868	88.00	89.61
xenon1	7.81	$4.25 \cdot 10^{-5}$	675	21.74	33.71
xenon2	28.28	$6.52 \cdot 10^{-5}$	707	21.69	33.60
venkat01	-	-	-	-	-
QC2534	-	-	-	-	-
mplate	-	-	-	-	-
waveguide3	132.87	$1.18 \cdot 10^{-5}$	10,107	4.59	7.58
ABACUS_shell_hd	882.64	0.68	92,773	0.31	0.51
light_in_tissue	4.42	$1.10 \cdot 10^{-5}$	355	26.90	43.91
kim1	66.03	$1.24 \cdot 10^{-5}$	4,128	73.98	75.10
chevron2	1,098.16	$1.95 \cdot 10^{-5}$	21,680	72.16	59.46

Table 3.30: Augmented AAR(6,12). Experiments without preconditioner.

	Augmented AAR$(m = 6,\, p = 12)$				
Matrix	Time (s)	rel.error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.02	$3.97 \cdot 10^{-9}$	25	69.69	89.42
fidapm37	14.66	$8.62 \cdot 10^{-3}$	5,711	16.84	20.55
raefsky5	0.04	$2.43 \cdot 10^{-9}$	25	79.08	-87.52
sp1	0.56	$2.12 \cdot 10^{-7}$	121	52.06	73.64
sp3	1.06	$3.58 \cdot 10^{-7}$	129	51.54	72.11
sp5	1.78	$3.27 \cdot 10^{-7}$	136	50.17	71.10
bcsstk29	-	-	-	-	-
sherman3	1.48	$1.56 \cdot 10^{-4}$	1,285	14.08	21.27
sherman5	0.16	$1.61 \cdot 10^{-6}$	260	35.32	54.05
fidap008	98.62	0.08	185,850	0.40	0.65
chipcool0	2.04	$1.01 \cdot 10^{-5}$	474	26.59	39.46
e20r0000	4.39	$2.64 \cdot 10^{-3}$	6,137	2.52	3.87
spmsrtls	-	-	-	-	-
III_Stokes	26.83	0.09	8,033	4.11	6.33
garon1	19.82	$1.19 \cdot 10^{-3}$	31,173	1.96	3.11
garon2	77.18	$4.49 \cdot 10^{-3}$	32,595	2.12	3.32
memplus	1.47	$5.10 \cdot 10^{-5}$	386	30.52	42.51
saylr4	1.35	$9.41 \cdot 10^{-3}$	2,471	13.43	20.35
xenon1	8.78	$4.06 \cdot 10^{-5}$	782	20.19	31.39
xenon2	33.39	$6.42 \cdot 10^{-5}$	818	20.80	31.49
venkat01	1,641.37	$8.36 \cdot 10^{-4}$	83,587	3.21	5.43
QC2534	-	-	-	-	-
mplate	-	-	-	-	-
waveguide3	158.63	$1.64 \cdot 10^{-5}$	11,813	4.41	6.98
ABACUS_shell_hd	-	-	-	-	-
light_in_tissue	4.80	$5.00 \cdot 10^{-5}$	360	31.55	45.62
kim1	3.05	$3.01 \cdot 10^{-7}$	134	52.38	71.46
chevron2	198.33	$1.20 \cdot 10^{-5}$	4,819	8.09	12.36

Table 3.31: Augmented AAR(6,12). Experiments with diagonal preconditioner.

an efficient acceleration for non-linear problems, studies have confirmed that it benefits also linear fixed point iterations. Standard approaches in this respect alternate a Richardson sweep and an Anderson mixing at each iteration as in Anderson-Richardson. All the variants of this scheme are comparable with either Full GMRES or Truncated GMRES. However, a drawback of these techniques is that they do not address issues such as the avoidance of global communications to leverage the performance on next generation computers. A recent step in this direction has been taken with the introduction of Alternating Anderson-Richardson. The idea behind this algorithm is to relax the frequency of Anderson mixings at periodic intervals, enhancing the computational locality and providing a path to an efficient parallelization. Theoretical equivalence between Full AAR and Full GMRES in exact arithmetic has been proved in this chapter. Numerical experiments in Section 3.3.5 show that truncated variants of AAR are competitive against truncated variants of AR, validating the claim that solving the least-squares problem at fewer iterations can benefit the computational time. Moreover, a study of the basis generated by AAR shows that the algorithm is more robust than AR against stagnation. This has been validated as well by ad-hoc numerical tests shown in Section 3.3.3. Results in Section 3.3.6 clearly show that AAR cannot compete with CG on symmetric positive definite systems. However, preliminary results on nonsymmetric or indefinite system with different choices of preconditioners suggest that AAR may be an appealing alternative to Restarted GMRES as to reduction of both time and global communications, as shown in numerical examples in Section 3.3.7. In Section 3.2.3 we introduced a new variant of AAR, called Augmented AAR. This variant has the advantage of guaranteeing convergence on linear systems with positive definite matrices. In addition, numerical experiments in Section 3.3.8 confirm that the augmented variant does not deteriorate the performance with respect to the unaugmented version introduced by P. Suryanarayana and collaborators. Therefore, it seems that Augmented AAR can be favored over Truncated AAR because of a more robust convergence analysis without negative impacts on the performance.

In the future a deeper mathematical analysis of AAR is needed, by exploring the properties of the projection subspaces used for the Anderson mixing and by assessing the performance sensitivity to quantities like relaxation parameters, periodic interval length and number of iterations involved in the mixing. Furthermore, a parallel implementation of the algorithm that could run on extreme scale machines is required, so that the possibility to reduce communication (leading to enhanced concurrency) and to require less memory than Restarted GMRES can be confirmed.

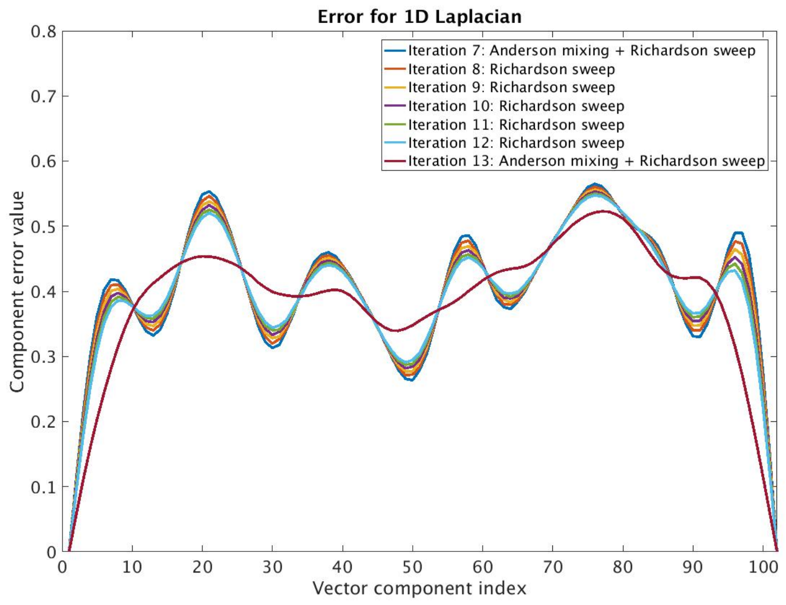

Figure 3.2: Error behavior for AAR solving a 1D Laplace problem without preconditioner. The iterations represented are the Richardson sweeps (Iterations from 8 to 12) and the successive Anderson mixing (Iteration 13).

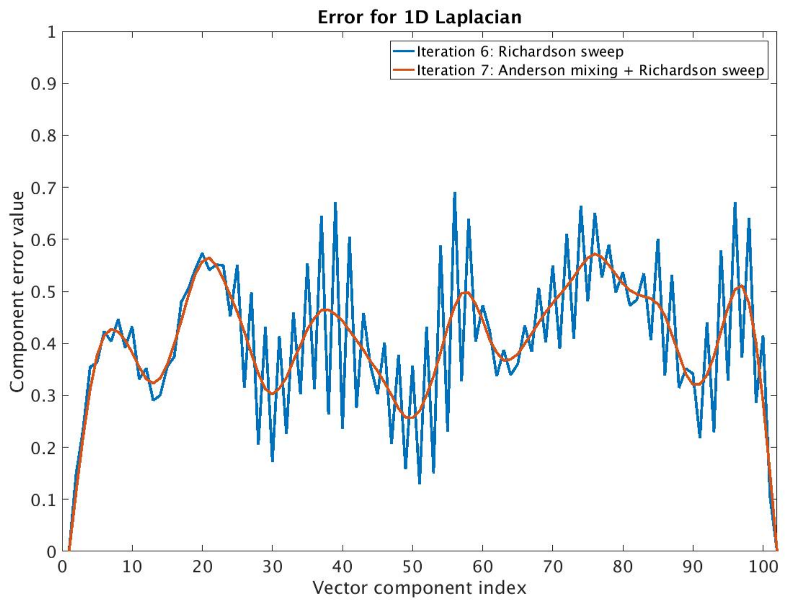

Figure 3.3: Error behavior for AAR solving a 1D Laplace without preconditioner. The iterations represented are the last of the first batch of Richardson sweeps and the first Anderson mixing.

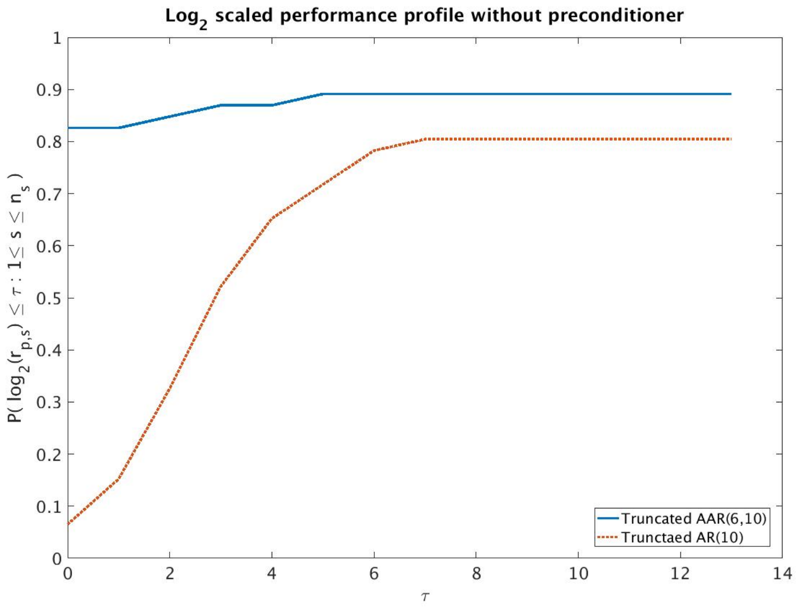

Figure 3.4: Performance profile in a $\log_2$ scale for Truncated AR(10) and Truncated AAR(6,10) on [0, 13]. Experiments without preconditioner.

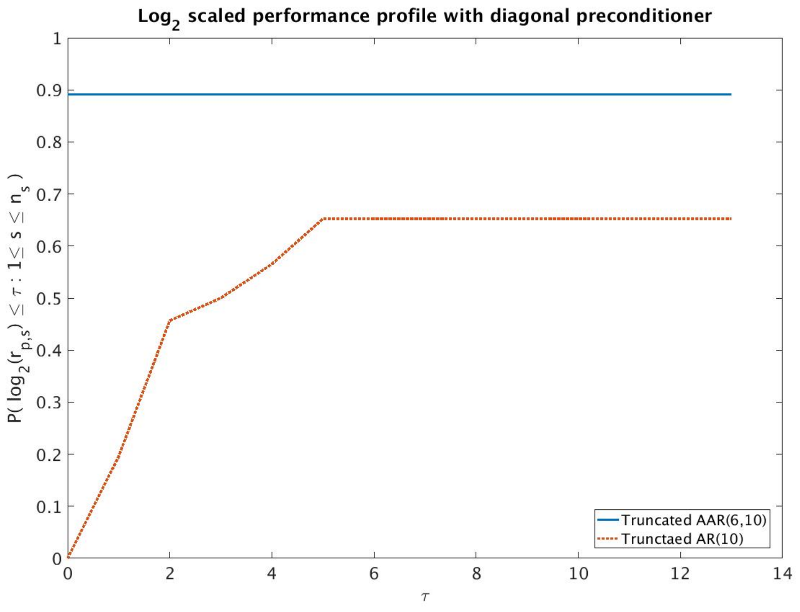

Figure 3.5: Performance profile in a $\log_2$ scale for Truncated AR(10) and Truncated AAR(6,10) on [0, 13]. Experiments with diagonal preconditioner.

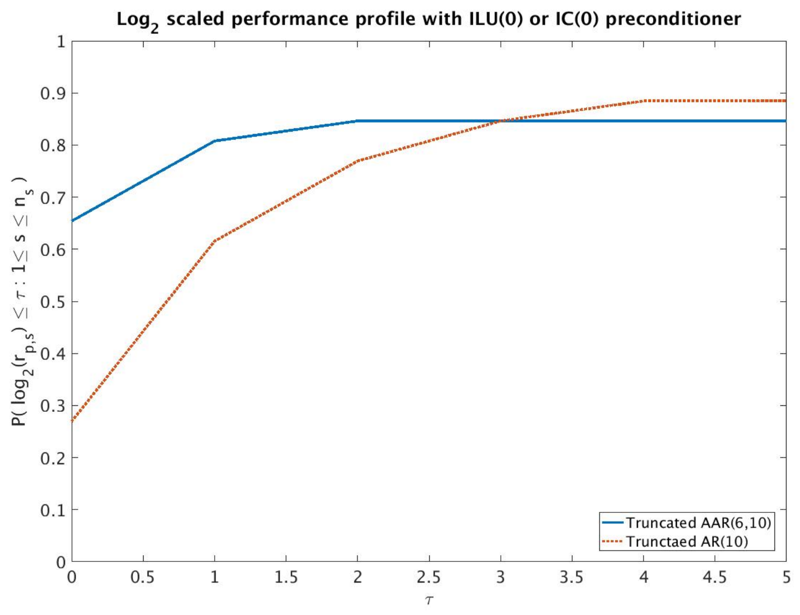

Figure 3.6: Performance profile in a $\log_2$ scale for Truncated AR(10) and Truncated AAR(6,10) on [0, 5]. Experiments with IC(0) preconditioner for real symmetric positive definite matrices and ILU(0) preconditioner for real matrices that are not symmetric positive definite.

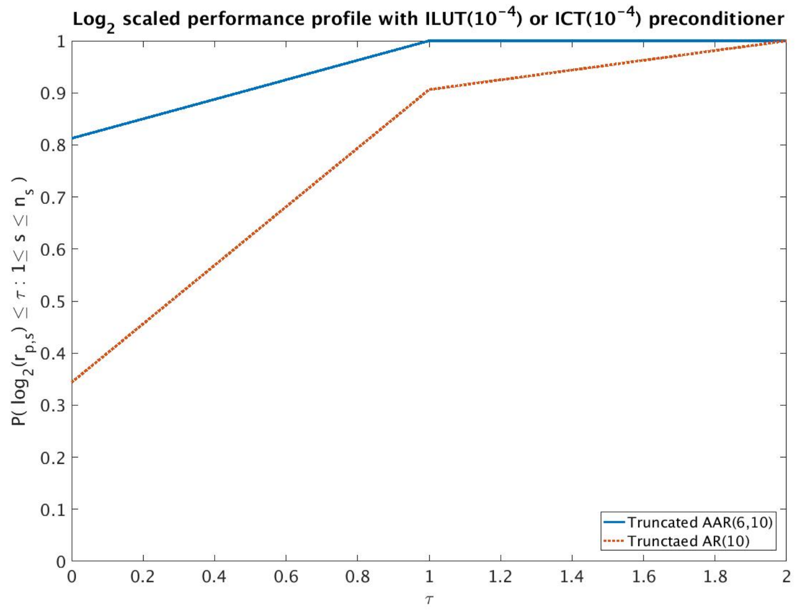

Figure 3.7: Performance profile in a $\log_2$ scale for Truncated AR(10) and Truncated AAR(6,10) on [0, 2]. Experiments with IC(10^{-4}) preconditioner for real symmetric positive definite matrices and ILUT(10^{-4}) preconditioner for real matrices that are not symmetric positive definite.

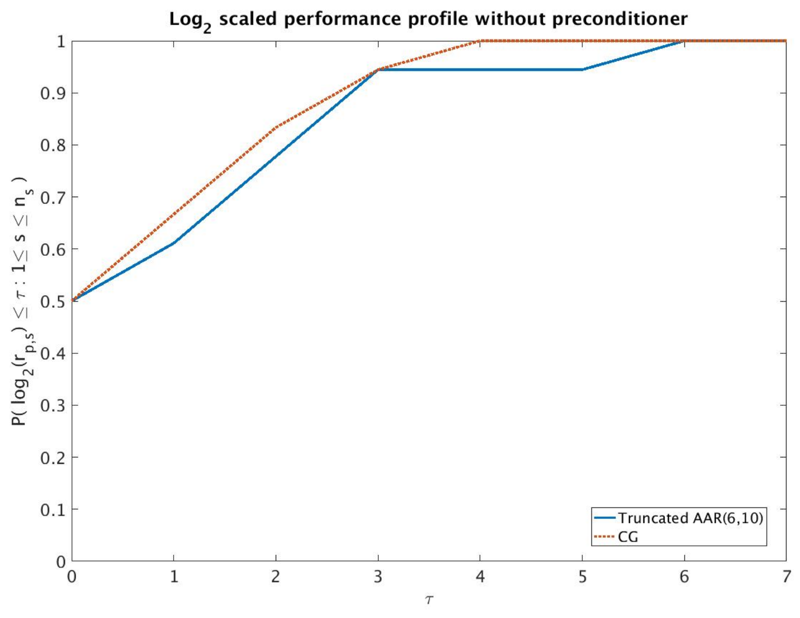

Figure 3.8: Performance profile in a $\log_2$ scale for Truncated AAR(6,10) and CG on [0, 7]. Real symmetric positive definite matrices. Experiments without preconditioner.

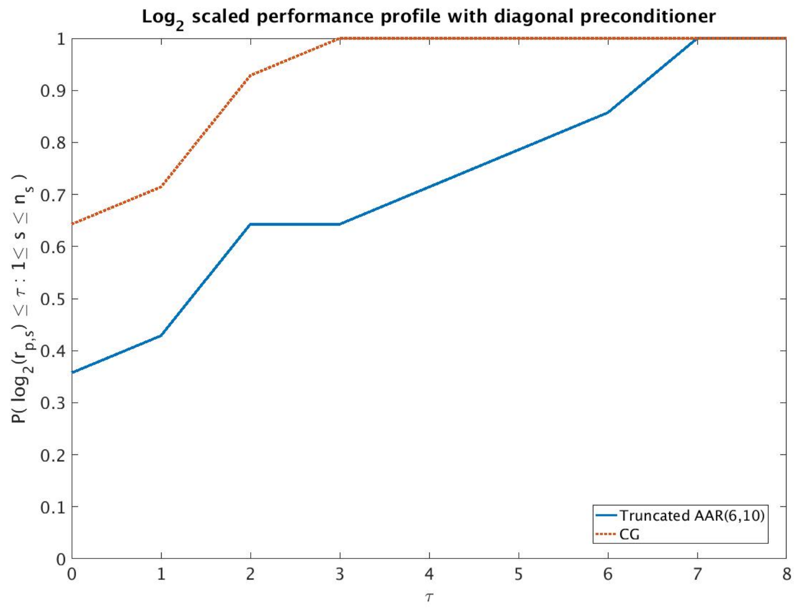

Figure 3.9: Performance profile in a $\log_2$ scale for Truncated AAR(6,10) and CG on [0, 8]. Real symmetric positive definite matrices. Experiments with diagonal preconditioner.

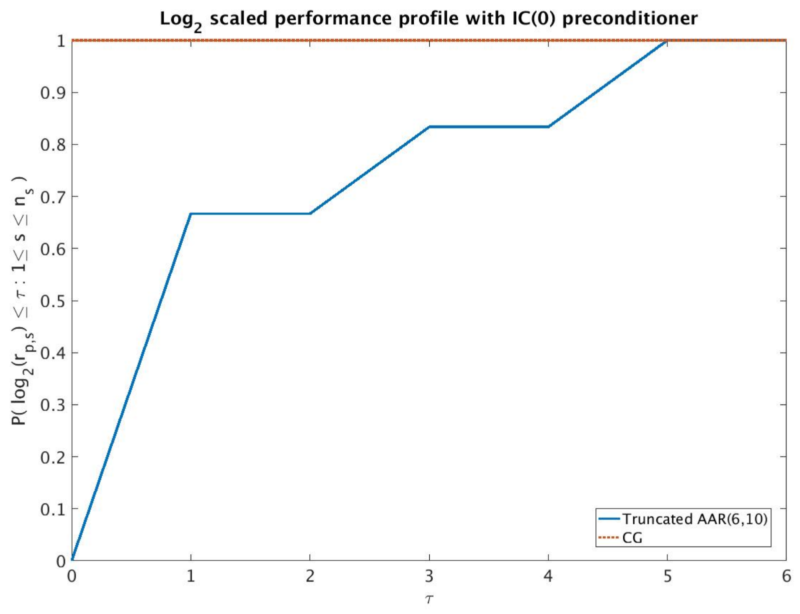

Figure 3.10: Performance profile in a $\log_2$ scale for Truncated AAR(6,10) and CG on [0, 6]. Real symmetric positive definite matrices. Experiments with IC(0) preconditioner.

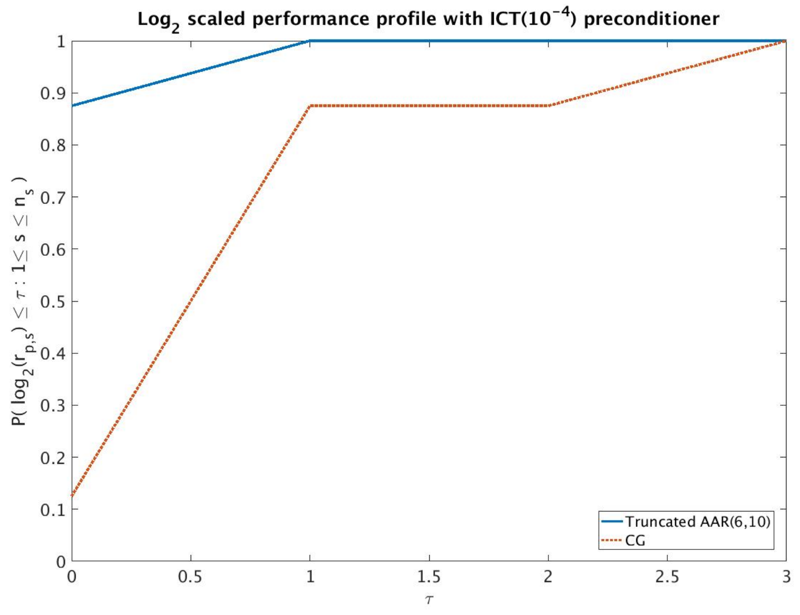

Figure 3.11: Performance profile in a $\log_2$ scale for Truncated AAR(6,10) and CG on [0, 3]. Real symmetric positive definite matrices. Experiments with ICT(10^{-4}) preconditioner.

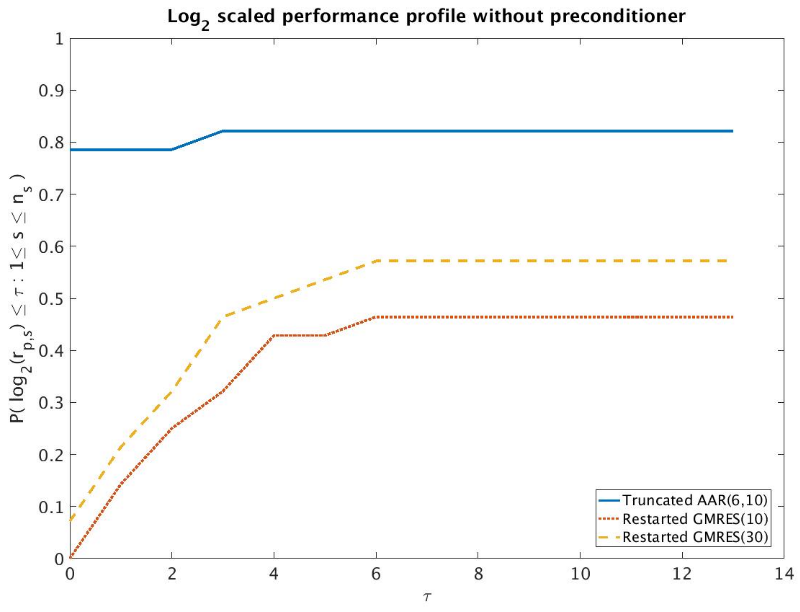

Figure 3.12: Performance profile in a $\log_2$ scale for Truncated AAR(6,10), Restarted GMRES(10) and Restarted GMRES(30) on [0, 13]. Experiments without preconditioner.

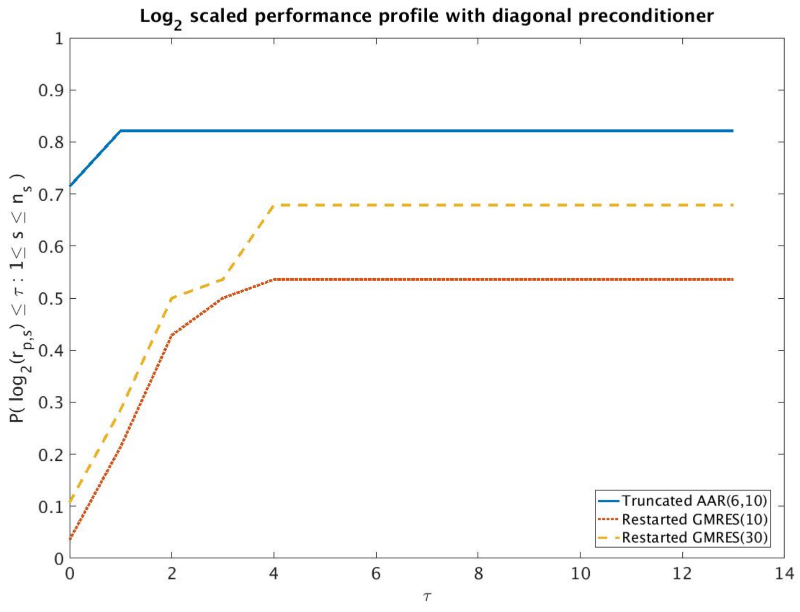

Figure 3.13: Performance profile in a $\log_2$ scale for Truncated AAR(6,10), Restarted GMRES(10) and Restarted GMRES(30) on [0, 13]. Experiments with diagonal preconditioner.

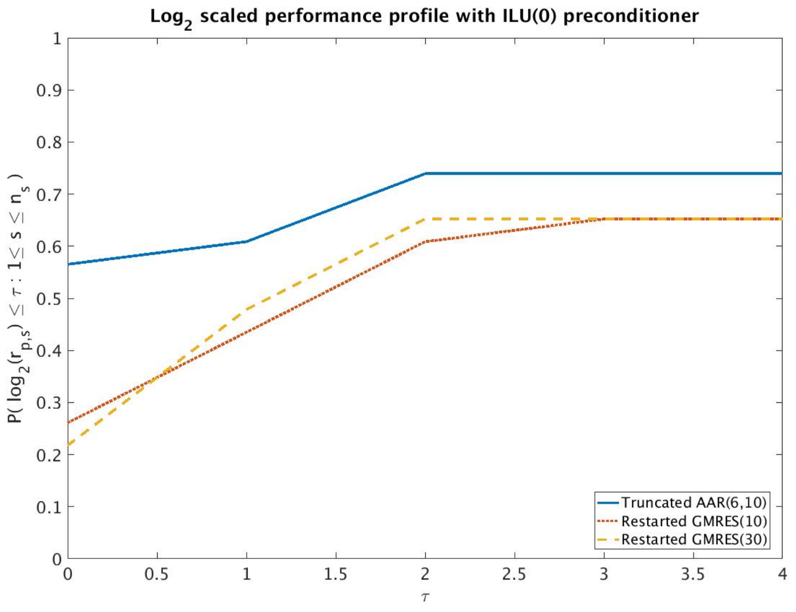

Figure 3.14: Performance profile in a $\log_2$ scale for Truncated AAR(6,10), Restarted GMRES(10) and Restarted GMRES(30) on [0, 4]. Experiments with ILU(0) preconditioner.

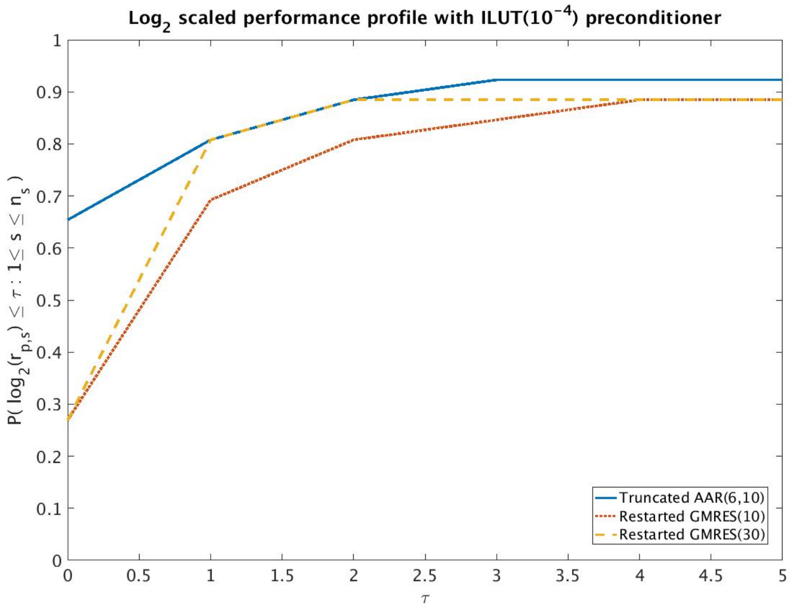

Figure 3.15: Performance profile in a $\log_2$ scale for Truncated AAR(6,10), Restarted GMRES(10) and Restarted GMRES(30) on [0, 5]. Experiments with ILUT(10^{-4}) preconditioner.

	Augmented AAR($m = 6$, $p = 12$)				
Matrix	Time (s)	rel.error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.01	$1.81 \cdot 10^{-12}$	13	1.99	89.99
fidapm37*	-	-	-	-	-
raefsky5*	0.06	$2.37 \cdot 10^{-11}$	7	89.43	-
sp1	0.22	$1.05 \cdot 10^{-8}$	43	70.49	89.30
sp3	0.40	$4.53 \cdot 10^{-8}$	43	68.96	89.41
sp5	0.69	$2.88 \cdot 10^{-8}$	43	68.80	88.20
bcsstk29*	$*$	$*$	$*$	$*$	$*$
sherman3	0.27	$5.00 \cdot 10^{-6}$	245	40.39	58.70
sherman5	0.06	$1.11 \cdot 10^{-7}$	62	61.01	85.54
fidap008*	-	-	-	-	-
chipcool0	0.59	$4.82 \cdot 10^{-7}$	109	53.37	75.16
e20r0000*	$*$	$*$	$*$	$*$	$*$
spmsrtls*	0.08	$6.55 \cdot 10^{-14}$	7	89.21	-
III_Stokes*	$*$	$*$	$*$	$*$	$*$
garon1*	$*$	$*$	$*$	$*$	$*$
garon2*	$*$	$*$	$*$	$*$	$*$
memplus	0.97	$1.15 \cdot 10^{-5}$	360	31.82	46.67
saylr4	0.13	$2.19 \cdot 10^{-6}$	110	52.78	76.20
xenon1*	-	-	-	-	-
xenon2*	-	-	-	-	-
venkat01	7.52	$7.97 \cdot 10^{-7}$	366	27.36	40.94
QC2534	334.36	$1.28 \cdot 10^{-5}$	93,915	1.27	2.08
mplate	179.32	6.91	22,657	1.62	2.54
waveguide3	206.15	$1.48 \cdot 10^{-6}$	13,225	5.94	8.59
ABACUS_shell_hd	-	-	-	-	-
light_in_tissue	1.93	$4.50 \cdot 10^{-7}$	112	53.26	73.84
kim1	0.33	$7.18 \cdot 10^{-9}$	14	46.85	89.31
chevron2	47.49	$8.74 \cdot 10^{-7}$	837	21.10	31.83

Table 3.32: Augmented AAR(6,12). Experiments with ILU(0) preconditioner.

	Augmented AAR$(m = 6,\ p = 12)$				
Matrix	Time (s)	rel.error	# Itr.	$\angle(\mathbf{r}^k, \mathbf{r}^{k+m})$	$\angle(\mathbf{r}^k, \mathbf{r}^{k+2m})$
fidap029	0.05	$6.34 \cdot 10^{-12}$	7	89.32	-
fidapm37*	-	-	-	-	-
raefsky5	0.06	$1.92 \cdot 10^{-11}$	7	88.12	-
sp1	0.11	$1.20 \cdot 10^{-9}$	7	89.02	-
sp3	0.14	$2.14 \cdot 10^{-9}$	7	87.35	-
sp5	0.22	$6.85 \cdot 10^{-9}$	8	83.47	-
bcsstk29*	$*$	$*$	$*$	$*$	$*$
sherman3	0.06	$7.63 \cdot 10^{-11}$	13	3.17	89.50
sherman5	0.05	$4.38 \cdot 10^{-11}$	13	2.58	89.94
fidap008*	-	-	-	-	-
chipcool0	0.36	$6.24 \cdot 10^{-9}$	16	47.34	89.75
e20r0000	0.15	$1.13 \cdot 10^{-5}$	24	59.51	34.65
spmsrtls	0.07	$5.24 \cdot 10^{-11}$	7	89.01	-
III_Stokes	0.85	$6.91 \cdot 10^{-5}$	29	4.91	3.02
garon1	0.13	$9.66 \cdot 10^{-9}$	19	30.09	22.39
garon2	0.64	$1.11 \cdot 10^{-7}$	34	44.70	50.64
memplus	0.09	$1.30 \cdot 10^{-10}$	13	3.65	89.96
saylr4	0.13	$1.37 \cdot 10^{-6}$	85	50.49	68.45
xenon1	0.92	$7.95 \cdot 10^{-10}$	25	60.71	87.71
xenon2	4.07	$5.68 \cdot 10^{-9}$	30	64.95	89.44
venkat01	1.98	$4.56 \cdot 10^{-10}$	19	65.39	88.47
QC2534	0.09	$1.11 \cdot 10^{-11}$	13	4.32	89.92
mplate	13.50	0.04	1,774	16.38	26.63
waveguide3	0.76	$6.16 \cdot 10^{-9}$	13	29.67	82.12
ABACUS_shell_hd	1.20	9.05	72	31.06	21.91
light_in_tissue	0.35	$7.31 \cdot 10^{-9}$	15	45.19	89.99
kim1	0.24	$3.87 \cdot 10^{-9}$	7	86.21	-
chevron2	4.54	$1.90 \cdot 10^{-8}$	55	52.50	63.94

Table 3.33: Augmented AAR(6,12). Experiments with ILUT(10^{-4}) preconditioner.

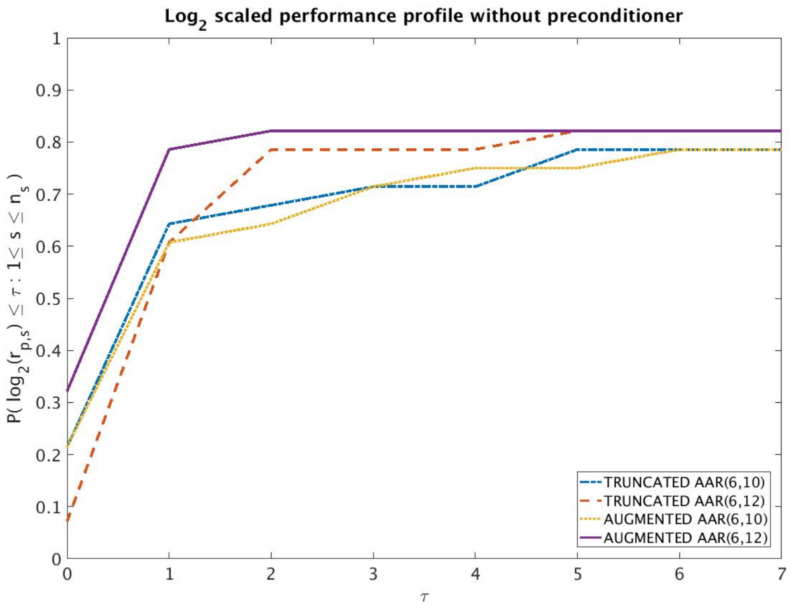

Figure 3.16: Performance profile in a $\log_2$ scale for Truncated AAR(6,10), Truncated AAR(6,12), Augmented AAR(6,10) and Augmented AAR(6,12) on [0, 7]. Experiments without preconditioner.

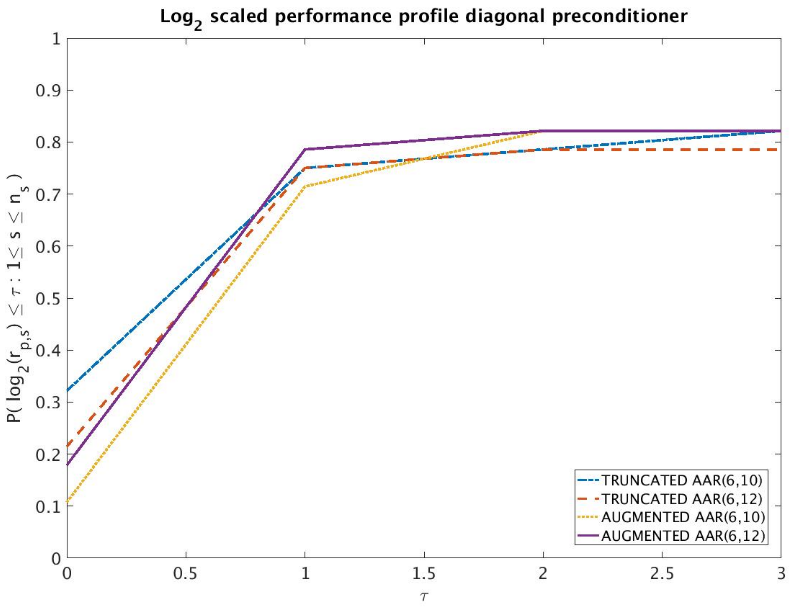

Figure 3.17: Performance profile in a $\log_2$ scale for Truncated AAR(6,10), Truncated AAR(6,12), Augmented AAR(6,10) and Augmented AAR(6,12) on [0, 3]. Experiments with diagonal preconditioner.

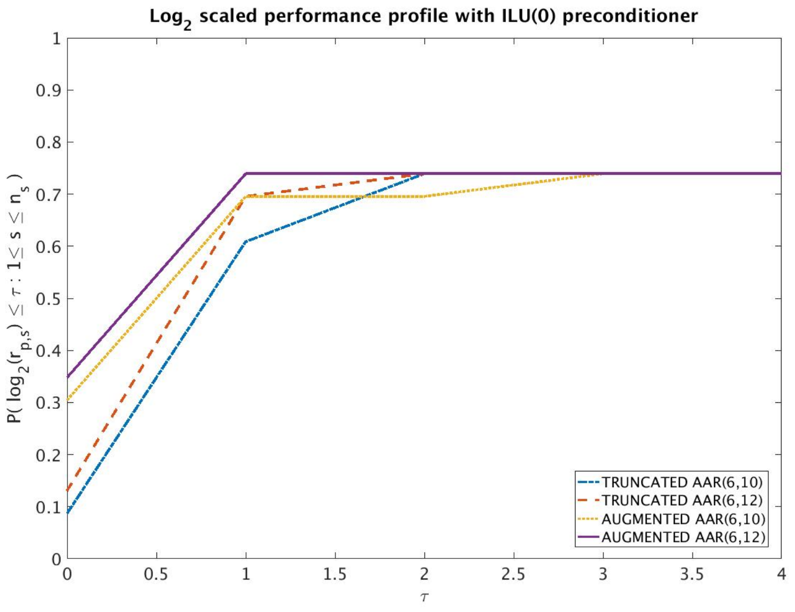

Figure 3.18: Performance profile in a $\log_2$ scale for Truncated AAR(6,10), Truncated AAR(6,12), Augmented AAR(6,10) and Augmented AAR(6,12) on [0, 4]. Experiments with ILU(0) preconditioner.

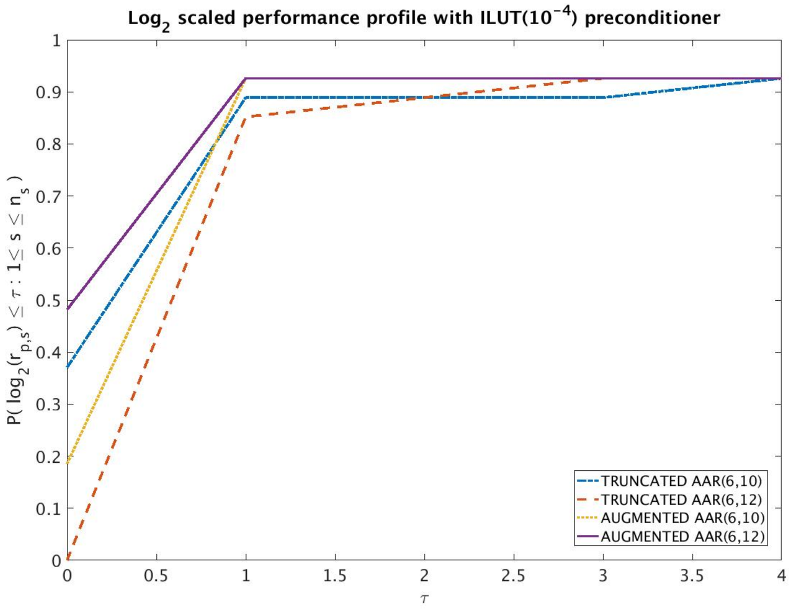

Figure 3.19: Performance profile in a $\log_2$ scale for Truncated AAR(6,10), Truncated AAR(6,12), Augmented AAR(6,10) and Augmented AAR(6,12) on [0, 4]. Experiments with ILUT(10^{-4}) preconditioner.

Chapter 4

Distributed memory parallelization of Alternating Anderson-Richardson

4.1 Introduction

This chapter assesses the performance of Alternating Anderson-Richardson in a multi-core computational environment. This analysis is conducted to motivate the use of this algorithm to solve large scale sparse linear systems in a High Performance Computing framework. Indeed, P. Suryanarayana and collaborators presented AAR as a linear solver suitable for next generation computing architectures due to the potential reduction of the inter-processor communication. Preliminary experiments in a multi-core environment have been shown by P. Suryanarayana and collaborators in [48] and [49]. In these works, the authors analyzed the performance of AAR to solve discretized elliptic partial differential equations described by a self-adjoint operator. Our goal is to extend this type of analysis to a broader class of linear systems, not necessarily focusing just on those arising from the discretization of PDEs.

We thus verify that the interprocessor communication can be significantly reduced using AAR in lieu of a standard Krylov method for linear systems associated with diverse applications, with a consequent benefit for the computational time. A *strong scaling* analysis of the algorithm is performed in this respect.

In Section 4.2 we describe the implementation of AAR used to run the experiments, followed by a description of the hardware adopted in Section 4.3. The numerical experiments conducted on a set of problems with different preconditioning techniques is described in Section 4.4. Section 4.5 analyzes the memory storage requirements for the different variants of AAR implemented, whereas the results of the experiments are discussed in Section 4.6. In conclusion, we comment the results based on the implementation of the code and we discuss possible future developments to improve the performance in Section 4.7.

4.2 Implementation in MPI

The code used to produce the numerical experiments of this chapter is written in `C` language. The library used to handle numerical linear algebra operations (e.g. storage of matrices, vectors, execution of matrix-vector multiplies and linear systems solving) is the `PETSc` library [66]. The matrix associated with the linear systems of interest is imported from a file written in binary format. The solution to the linear system is generated as a random vector and the right-hand side vector is constructed multiplying the solution vector by the coefficient matrix. The least-squares problems in Algorithm 6 of Chapter 3 to compute the Anderson mixing are solved according to two different approaches. Indeed, a version of AAR solves the least-squares problem via LSQR [46], whereas another version explicitly builds the normal equation

$$(R_k^T R_k)\mathbf{g}^k = R_k^T \mathbf{r}^k. \tag{4.1}$$

The normal equation (4.1) is solved independently by each processor computing the pseudo-inverse of the matrix $R_k^T R_k$. This task is accomplished using the `LAPACKE_dgelsd` routine inside the `LAPACK` library [64]. Although we are aware that the ill-conditioning of the least-squares problem can be severely affected by explicitly building the normal equation (4.1), this seems to be the most convenient option to carry out the project in the short term. These two approaches were the only feasible ones according to the software resources provided. Indeed, `PETSc` does not currently allow one to compute the QR-factorization of a rectangular matrix. Moreover, the options available were restricted to the libraries installed on the cluster, preventing us from linking external libraries (e.g. `SCALAPACK`). Although the numerical properties of Equation (4.1) can negatively impact the accuracy of the least-squares solving, we remind the reader that this drawback can be controlled by combining the use of AAR with an efficient preconditioner. Moreover, the explicit construction of the normal equation has appealing properties from a parallelism perspective. Indeed, the data structures needed for linear algebra operations such as matrices and vectors are distributed row-wise across the `MPI` processes. For the sake of simplicity, let us temporarily assume that the number of rows is a multiple of the number of the `MPI` processes instantiated and that the number of rows owned by each process is equal to n_{loc}. If an `MPI` process with ID equal to b owns rows from index $[(b-1)\cdot n_{\text{loc}}+1]$ to $[b\cdot n_{\text{loc}}]$, then each process can locally compute

$$\left[(R_{k,b}^{\text{loc}})^T R_{k,b}^{\text{loc}}\right] \quad \text{and} \quad (R_{k,b}^{\text{loc}})^T \mathbf{r}_{\text{loc},b}^k, \tag{4.2}$$

where $R_{k,b}^{\text{loc}}$ is the submatrix of R_k obtained by extracting the rows with indices from $[(b-1)\cdot n_{\text{loc}}+1]$ to $[b\cdot n_{\text{loc}}]$ and $\mathbf{r}_{\text{loc},b}^k$ is the subvector of $\mathbf{r}^k$ with indices from $[(b-1)\cdot n_{\text{loc}}+1]$ to $[b\cdot n_{\text{loc}}]$. The matrices $\left[(R_{k,b}^{\text{loc}})^T R_{k,b}^{\text{loc}}\right]$ and $(R_{k,b}^{\text{loc}})^T \mathbf{r}_{\text{loc},b}^k$ are computed independently by each process without any interprocessor communication needed.

The only interaction among processes is needed to reconstruct the global quantities to obtain Equation (4.1). In particular, only an `MPI_Allreduce` operation is employed using the sum as global reduction operation to reconstruct $(R_k^T R_k)$ and $R_k^T \mathbf{r}^k$ as follows:

$$(R_k^T R_k) = \sum_{b=1}^{n_{\text{proc}}} \left[(R_{k,b}^{\text{loc}})^T R_{k,b}^{\text{loc}} \right], \quad R_k^T \mathbf{r}^k = \sum_{b=1}^{n_{\text{proc}}} (R_{k,b}^{\text{loc}})^T \mathbf{r}_{\text{loc},b}^k,$$

where n_{proc} is the total number of `MPI` processes instantiated. Once $(R_k^T R_k)$ and $R_k^T \mathbf{r}^k$ are reconstructed and locally stored on each processor, the solution to the least-squares problem is locally computed by solving Equation (4.1). We recall that if Truncated AAR(m,p) is used, then the matrix $(R_k^T R_k)$ has size $p \times p$, and generally p does not exceed 10. Therefore, the normal equation that has to be solved sequentially by each process is computationally cheap. By adopting this approach to compute the vector $\mathbf{g}^k$ for the Anderson mixing, we minimize the amount of global communications needed to execute the Algorithm 6. Indeed, the only global communication needed is the reducing sum for the construction of the matrix $(R_k^T R_k)$ and of the vector $R_k^T \mathbf{r}^k$. Therefore, if a proper preconditioner is combined with AAR in order to cap the ill-conditioning of the normal equation, this implementation is supposed to considerably leverage the parallelism in a multi-core environment with respect to standard Krylov methods.

4.3 Hardware configuration

The numerical experiments are run on the `mps` queue provided by the Partnership for an Advanced Computing Environment (PACE) at Georgia Institute of Technology. The `mps` queue provides access to a 23 node cluster featuring 64 cores (4x AMD Opteron(tm) Processor 6378) and 128 GB RAM per node, with a memory per core ratio of 2GB. In total, the cluster offers 1,472 cores interconnected with QDR InfiniBand for a total of almost 3 terabytes of RAM. However, not all the 23 nodes composing the `mps` queue have the same hardware features. Because of this, we employ only the 16 nodes which share the following configuration: Altus 1804i Server - 4P Interlagos Node, Quad AMD Opteron 6276, 16C, 2.3 GHz, 128GB, DDR3- 1333 ECC, 80GB SSD, MLC, 2.5" HCA, Mellanox ConnectX 2, 1-port QSFP, QDR, memfree, CentOS, Version 5, and connected through InfiniBand cable.

4.4 Preconditioning approaches

The `PETSc` library provides several options as preconditioners. However, only some of them can be run in a parallel computational environment. The ones we focus on for the numerical experiments of this chapter are:

- Block Incomplete LU factorization with zero fill-in (Block-ILU(0) for short)
- Restricted Additive Schwarz (RAS for short)
- Incomplete LU factorization with thresholding, (ILUT(τ) for short)
- Sparse Approximate Inverse (SPAI for short)
- Algebraic Multigrid (AMG for short)

As concerns Block-ILU(0) and RAS preconditioners, the total number of subdomains is set to 1,024. This choice is dictated by the fact that 1,024 is the maximal number of `MPI` processes that can be instantiated on the cluster used for the experiments.

With regards to the RAS preconditioner, ILU(0) is used as a local solver on each subdomain. The subdomain solver adopted for RAS is ILU(0).

With regards to the SPAI preconditioner, the implementation provided in `PETSc` is based on the method described in [30]. The threshold to control the fill-in in ILUT(τ) is set to $\tau = 10^{-4}$, whereas the threshold to control the fill-in in SPAI preconditioners is set to 0.01 with a number of levels equal to 2. The number of levels coincide with the power of the coefficient matrix adopted to control the sparsity pattern.

The algebraic multigrid preconditioner is constructed so that a V-cycle is adopted at each application of the preconditioner. The total number of levels in the multigrid hierarchy is set to 5 and the coarsening factor is equal to 3. Chebyshev smoothers are adopted at each level and a direct solver is employed at the coarse level. Unsmoothed aggregation is applied to transfer the residual across the hierarchy.

4.5 Memory storage requirements

As applications in science and engineering are increasing at scale, memory requirements are becoming an issue. Indeed, state-of-the-art computers are progressively optimized in terms of performance (i.e. increasing FLOPS), but the manufacturers have reached a memory cap due to space limitations in the transistors. Therefore, the memory requirements of future applications are supposed to challenge current memory capabilities. Because of this, new linear solvers must be conscientiously developed in order to minimize the memory storage needed to perform algebraic operations. In this context we quantify the memory storage using the scalar value stored in double precision as a unit.

As concerns solving the least-squares problem using the LSQR algorithm, the process resorts to the Golub-Kahan bidiagonalization technique (GKB for short). Therefore, at each iteration of the GKB algorithm, an approximate factorization of R_k is computed as:

$$R_k \approx U_k B_k V_k^T,$$

where $B_k \in \mathbb{R}^{k\times k}$, $U_k \in \mathbb{R}^{n\times k}$ and $V_k \in \mathbb{R}^{p\times k}$. The matrices U_k and V_k have orthogonal columns. In a least-squares framework, the matrix U_k does not need to be stored. Therefore, only B_k and V_k are needed. The memory requirement associated with the

GKB algorithm at the kth iteration amounts to

$$\text{memory storage required by GKB at the } k\text{th iteration} = k(2+p).$$

When LSQR is used to solve the least-squares problem, the maximum number of iterations allowed is set equal to the number of columns of R_k, that is p. Therefore, in the worst case scenario where p iterations are performed, the maximum memory storage needed is

$$\text{memory storage required by GKB at the } p\text{th iteration} = p(2+p).$$

This amount of memory can be equally split among all the processes employed.

As concerns solving the least-squares problem computing the SVD on the matrix $R_k^T R_k$, the goal is to decompose the matrix $R_k^T R_k$ as

$$R_k^T R_K = W_k \Sigma_k Z_k^T,$$

where $\Sigma_k \in \mathbb{R}^{p\times p}$ is a diagonal matrix, whereas $W_k \in \mathbb{R}^{p\times p}$ and $Z_k \in \mathbb{R}^{p\times p}$ are orthogonal. In this case, the total amount of memory required on each processor is

$$\text{memory storage required by SVD computed on each processor} = p(2p+1).$$

Therefore, the memory required on each processor by the AAR algorithm using the SVD is bigger than the one required by the AAR using LSQR to solve the least-square problem. However, we remind the reader that p is at the order of $\mathcal{O}(10)$. Hence the memory storage is still affordable.

4.6 Numerical experiments

The numerical experiments aim to compare the performance of Restarted GMRES(10) and Restarted GMRES(30) with two different implementations of Truncated AAR(6,10). The first implementation uses the `PETSc` version of LSQR to solve the least-squares problem, whereas the second implementation of AAR explicitly builds the normal equation and solves it on each processor via the SVD decomposition. When the LSQR is adopted to iteratively compute the Anderson mixing vector $\mathbf{g}^k$, the maximum number of iterations allowed is set to p, i.e. the number of columns of R_k.

The test cases have been taken from the SuiteSparse Matrix Collection [18] and are described in Table 4.1. The experiments are run for no use of the preconditioner, Block-ILU(0), Restricted Additive Schwarz, ILUT(10^{-4}), SPAI and Algebraic Multigrid. The specifics about the set-up of the preconditioners are as described in Section 4.4. All the preconditioners are applied as left-preconditioners. The stopping criterion requires that the relative ℓ^2-norm of the preconditioned residual be less than 10^{-8} and the maximum number of iterations allowed is set to 10^5. The results averaged over 5 runs. For each run, the solution vector has been generated as a random vector and the right hand side has been computed via the matrix vector multiply between the coefficient matrix and the solution vector. The time limit for the simulations to run and complete their task (i.e. construction of the preconditioner and execution of 5 runs per linear solver) was set to 7 CPU hours. The numerical experiments have been run for 32, 64, 128, 256 and 512 CPU cores. The option with 1,024 CPU cores has not been explored because communication overhead already impacted the performance of the linear solvers for 512 CPU cores, as shown by the tables described as follows.

The tables include only those cases where the preconditioner could be constructed in a numerically stable way and that allowed at least one of the linear solvers to converge. The use of the world "Overtime" that sometimes is used in the Tables refers to situations where the construction of the preconditioner took longer than two hours.

The results for Block-ILU(0) are displayed in Table 4.2. The Block-ILU(0) preconditioner cannot be constructed in a numerically stable way for the matrices `Freescale1` and `FullChip` and for the matrix `dielFilterV2real` none of the linear solvers considered converges to the prescribed accuracy. For the remaining test cases where convergence has been achieved, it can be noticed that the implementation of AAR using LSQR is not competitive against Restarted GMRES. This can be mainly explained by the fact that LSQR itself employs inner products to iteratively compute the solution to a least-squares problem. Therefore, LSQR is very likely to cause bottlenecks for the parallelization by introducing significant communication overheads. However, the AAR implementation that resort the the normal equation exhibits very promising results. Indeed, the computational locality of the preconditioner accommodates the strong scalability of the overall iterative procedure up to 256 cores. In this context, AAR using the normal equation performs better than Restarted GMRES(10) and Restarted GMRES(30). The performance deteriorates going up to 512 cores because of non-negligible communication overheads.

In Table 4.3 we report the results for the use of Restricted Additive Schwarz. In general, the use of a RAS as a preconditioner rather than Block-ILU(0) should improve the performance of the linear solver, since an overlap across the subdomain generally facilitates the convergence. However, this does not occur for the matrix `CurlCurl4`. Indeed, RAS does not allow convergence for this test case on any of the linear solvers tested. Analogously, the RAS preconditioner does not allow convergence on the matrices `Freescale1`, `FullChip` and `dielFilterV2real`. As for the remaining test cases, we still have Truncated AAR(6,10) with LSQR perform worse than any implementation of Restarted GMRES, whereas Truncated AAR(6,10) with the normal equation performs comparably to Restarted GMRES for `atmosmodl` and `circuit5M_dc` and it clearly outperforms Restarted GMRES on the matrix `Transport`.

As concerns ILUT(10^{-4}), results are displayed in Table 4.4. The construction of the preconditioner exceeded two hours for the matrices `circuit5M_dc` and `dielFilterV2real`, leading us to interrupt the analysis for these test cases. Moreover, none of the linear solvers attained the prescribed accuracy on the matrices `Freescale1` and `FullChip`. As concerns the remaining matrices, the performance of the linear solvers relative to each other is similar to what experienced for the previous preconditioning techniques. Indeed, Truncated AAR(6,10) using LSQR is the linear solver that performed the worst, whereas Truncated AAR with the normal equation converged faster than Restarted GMRES(10) and Restarted GMRES(30). The performance of the linear solvers for the matrices `CurlCurl4` and `atmosmodl` is not reported for 512 processes because the construction of the preconditioner exceeded the time allowed of two hours.

The results for the use of SPAI as a preconditioner are shown in Table 4.5. The set-up phase of the preconditioner did not finish in two hours for `dielFilterV2real`. As concerns `Freescale1`, `FullChip` and `CurlCurl4` instead, none of the linear solvers achieved convergence. With regards to the other test cases, we still have Truncated AAR(6,10) with LSQR take longer than any other linear solver, whereas Truncated AAR(6,10) with the normal equation succeeds in outperforming both the versions of Restarted GMRES tested.

The use of Algebraic Multigrid preconditioners lead to results in Table 4.6. The construction of the preconditioner brokedown for `circuit5M_dc`, whereas it exceeded two hours for `Freescale1` and `FullChip`. The algebraic multigrid preconditioner is the only approach that makes the linear solvers converge with `dielFilterV2real`. For this test case, the version of Truncated AAR(6,10) using the normal equation is still the linear solver that performs the best. However, for the matrices `atmosmodl` and `Transport` the situation changes. Indeed, the two versions of Restarted GMRES converge faster than both the implementations of AAR for these problems.

Matrix	Type	Size	Structure	Positive definite	Field of application
atmosmodl	real	1,489,752	nonsymmetric	no	computational fluid dynamics
circuit5M_dc	real	3,523,317	nonsymmetric	no	circuit simulation
Freescale1	real	3,428,755	nonsymmetric	no	circuit simulation
FullChip	real	2,987,012	nonsymmetric	no	circuit simulation
CurlCurl_4	real	2,380,515	symmetric	no	model reduction problem
dielFilterV2real	real	1,157,456	symmetric	no	electromagnetics
Transport	real	1,602,111	nonsymmetric	no	structural engineering

Table 4.1: `MPI` experiments. List of matrices used for numerical experiments.

Block-ILU(0) preconditioner					
atmosmodl	**Number of processes**				
Linear solver	**32**	**64**	**128**	**256**	**512**
TR_AAR_LSQR(6,10)	18.48 (s)	10.00 (s)	4.28 (s)	6.27 (s)	19.03 (s)
TR_AAR_NE(6,10)	5.87 (s)	2.71 (s)	1.28 (s)	0.80 (s)	0.72 (s)
Restarted GMRES(10)	4.99 (s)	2.52 (s)	1.24 (s)	0.78 (s)	0.70 (s)
Restarted GMRES(30)	4.98 (s)	2.29 (s)	1.27 (s)	0.70 (s)	0.59 (s)
circuit5M_dc	**Number of processes**				
Linear solver	**32**	**64**	**128**	**256**	**512**
TR_AAR_LSQR(6,10)	12.62 (s)	7.49 (s)	4.30 (s)	3.91 (s)	5.81 (s)
TR_AAR_NE(6,10)	4.90 (s)	2.63 (s)	1.51 (s)	1.35 (s)	1.30 (s)
Restarted GMRES(10)	3.11 (s)	1.83 (s)	1.31 (s)	1.34 (s)	1.37 (s)
Restarted GMRES(30)	3.12 (s)	1.82 (s)	1.40 (s)	1.38 (s)	1.39 (s)
Freescale1(RCM reordering)	**Number of processes**				
Linear solver	**32**	**64**	**128**	**256**	**512**
TR_AAR_LSQR(6,10)	1,865.75 (s)	967.27 (s)	613.60 (s)	663.63 (s)	1,009.82 (s)
TR_AAR_NE(6,10)	757.23 (s)	378.20 (s)	243.45 (s)	254.71 (s)	270.59 (s)
Restarted GMRES(10)	-	-	-	-	-
Restarted GMRES(30)	-	-	-	-	-
CurlCurl4	**Number of processes**				
Linear solver	**32**	**64**	**128**	**256**	**512**
TR_AAR_LSQR(6,10)	270.83 (s)	141.87 (s)	81.53 (s)	66.38 (s)	160.55 (s)
TR_AAR_NE(6,10)	107.59 (s)	51.41 (s)	26.31 (s)	15.42 (s)	14.35 (s)
Restarted GMRES(10)	466.85 (s)	261.19 (s)	125.71 (s)	97.47 (s)	88.43 (s)
Restarted GMRES(30)	459.79 (s)	260.29 (s)	126.22 (s)	97.11 (s)	91.23 (s)
Transport	**Number of processes**				
Linear solver	**32**	**64**	**128**	**256**	**512**
TR_AAR_LSQR(6,10)	334.20 (s)	180.32 (s)	94.88 (s)	107.32 (s)	310.34 (s)
TR_AAR_NE(6,10)	121.61 (s)	59.48 (s)	34.60 (s)	22.26 (s)	40.70 (s)
Restarted GMRES(10)	590.42 (s)	270.52 (s)	158.32 (s)	121.90 (s)	171.39 (s)
Restarted GMRES(30)	591.14 (s)	269.04 (s)	163.27 (s)	121.09 (s)	167.50 (s)

Table 4.2: `MPI` experiments. Block-ILU(0) preconditioner.

Restricted Additive Schwarz					
atmosmodl	**Number of processes**				
Linear solver	**32**	**64**	**128**	**256**	**512**
TR_AAR_LSQR(6,10)	11.63 (s)	6.58 (s)	3.65 (s)	3.80 (s)	8.24 (s)
TR_AAR_NE(6,10)	5.20 (s)	3.01 (s)	1.69 (s)	0.89 (s)	0.57 (s)
Restarted GMRES(10)	4.00 (s)	2.52 (s)	1.48 (s)	0.89 (s)	0.67 (s)
Restarted GMRES(30)	4.02 (s)	2.57 (s)	1.39 (s)	0.82 (s)	0.79 (s)
circuit5M_dc	**Number of processes**				
Linear solver	**32**	**64**	**128**	**256**	**512**
TR_AAR_LSQR(6,10)	8.02 (s)	4.61 (s)	2.94 (s)	2.72 (s)	3.59 (s)
TR_AAR_NE(6,10)	3.56 (s)	1.92 (s)	1.24 (s)	1.17 (s)	1.09 (s)
Restarted GMRES(10)	2.20 (s)	1.27 (s)	0.97 (s)	0.90 (s)	0.89 (s)
Restarted GMRES(30)	2.21 (s)	1.29 (s)	0.90 (s)	0.90 (s)	0.86 (s)
Transport	**Number of processes**				
Linear solver	**32**	**64**	**128**	**256**	**512**
TR_AAR_LSQR(6,10)	140.42 (s)	78.68 (s)	35.04 (s)	36.65 (s)	67.08 (s)
TR_AAR_NE(6,10)	75.29 (s)	39.54 (s)	24.80 (s)	14.59 (s)	9.58 (s)
Restarted GMRES(10)	155.79 (s)	74.23 (s)	53.71 (s)	30.38 (s)	27.65 (s)
Restarted GMRES(30)	155.74 (s)	74.10 (s)	53.34 (s)	31.91 (s)	21.28 (s)

Table 4.3: `MPI` experiments. Restricted Additive Schwarz preconditioner with overlap layer equal to 1.

Incomplete LU factorization with Thresholding					
atmosmodl	**Number of processes**				
Linear solver	**32**	**64**	**128**	**256**	**512**
TR_AAR_LSQR(6,10)	20.01 (s)	14.43 (s)	6.76 (s)	6.34 (s)	Overtime
TR_AAR_NE(6,10)	6.83 (s)	5.90 (s)	2.81 (s)	1.63 (s)	Overtime
Restarted GMRES(10)	5.14 (s)	8.19 (s)	4.86 (s)	1.96 (s)	Overtime
Restarted GMRES(30)	5.13 (s)	8.24 (s)	4.60 (s)	2.04 (s)	Overtime
CurlCurl4	**Number of processes**				
Linear solver	**32**	**64**	**128**	**256**	**512**
TR_AAR_LSQR(6,10)	1,207.72 (s)	1,523.01 (s)	797.01 (s)	588.11 (s)	Overtime
TR_AAR_NE(6,10)	428.23 (s)	383.26 (s)	189.23 (s)	121.32 (s)	Overtime
Restarted GMRES(10)	779.39 (s)	330.43 (s)	205.56 (s)	152.88 (s)	Overtime
Restarted GMRES(30)	779.43 (s)	329.84 (s)	207.04 (s)	151.33 (s)	Overtime
Transport	**Number of processes**				
Linear solver	**32**	**64**	**128**	**256**	**512**
TR_AAR_LSQR(6,10)	144.38 (s)	78.68 (s)	35.04 (s)	36.65 (s)	67.08 (s)
TR_AAR_NE(6,10)	100.50 (s)	39.54 (s)	24.80 (s)	14.59 (s)	9.58 (s)
Restarted GMRES(10)	142.81 (s)	74.23 (s)	53.71 (s)	30.38 (s)	27.65 (s)
Restarted GMRES(30)	142.65 (s)	74.10 (s)	53.34 (s)	31.91 (s)	21.28 (s)

Table 4.4: `MPI` experiments. ILUT(10^{-4}) preconditioner.

Sparse Approximate Inverse					
atmosmodl	**Number of processes**				
Linear solver	**32**	**64**	**128**	**256**	**512**
TR_AAR_LSQR(6,10)	11.70 (s)	6.18 (s)	3.07 (s)	3.88 (s)	8.74 (s)
TR_AAR_NE(6,10)	5.09 (s)	2.61 (s)	1.40 (s)	0.90 (s)	0.84 (s)
Restarted GMRES(10)	3.84 (s)	2.04 (s)	1.11 (s)	0.67 (s)	0.79 (s)
Restarted GMRES(30)	3.83 (s)	1.97 (s)	1.11 (s)	0.74 (s)	0.86 (s)
circuit5M_dc	**Number of processes**				
Linear solver	**32**	**64**	**128**	**256**	**512**
TR_AAR_LSQR(6,10)	3.97 (s)	2.64 (s)	1.37 (s)	1.25 (s)	2.28 (s)
TR_AAR_NE(6,10)	1.37 (s)	0.71 (s)	0.48 (s)	0.46 (s)	0.60 (s)
Restarted GMRES(10)	0.65 (s)	0.36 (s)	0.29 (s)	0.30 (s)	0.56 (s)
Restarted GMRES(30)	0.65 (s)	0.38 (s)	0.28 (s)	0.28 (s)	0.59 (s)
Transport	**Number of processes**				
Linear solver	**32**	**64**	**128**	**256**	**512**
TR_AAR_LSQR(6,10)	128.14 (s)	88.88 (s)	45.31 (s)	52.27 (s)	136.83 (s)
TR_AAR_NE(6,10)	63.95 (s)	29.23 (s)	16.27 (s)	9.95 (s)	16.35 (s)
Restarted GMRES(10)	157.93 (s)	87.76 (s)	52.60 (s)	36.17 (s)	80.85 (s)
Restarted GMRES(30)	157.79 (s)	86.89 (s)	51.53 (s)	38.97 (s)	95.92 (s)

Table 4.5: `MPI` experiments. SPAI(0.01) preconditioner with 2 levels.

4.7 Conclusions and future work

In this chapter we have described two different ways to implement AAR in `C` using the `PETSc` library as a support to perform linear algebra operations. One possibility allowed by the routines in `PETSc` is to solve the least-squares problem for the Anderson mixing using the iterative method LSQR. The other approach presented requires explicitly building the normal equation. Despite the explicit construction of the normal equation can lead to ill-conditioned leas-squares problems, it has appealing computational properties from a parallelization perspective. Indeed, the use of the normal equation to solve the least-squares problem would minimize the global communication required across the MPI processes. Therefore, the concurrency of the algorithm may be significantly increased. This is confirmed by the numerical results obtained. In fact, the implementation of Truncated AAR(6,10) with the normal equation generally outperforms all the other linear solvers tested. For all the precon-

Algebraic Multigrid					
atmosmodl	**Number of processes**				
Linear solver	**32**	**64**	**128**	**256**	**512**
TR_AAR_LSQR(6,10)	3.50 (s)	2.02 (s)	1.41 (s)	1.37 (s)	3.46 (s)
TR_AAR_NE(6,10)	2.97 (s)	1.72 (s)	1.11 (s)	1.24 (s)	2.67 (s)
Restarted GMRES(10)	1.16 (s)	0.78 (s)	0.45 (s)	0.48 (s)	1.19 (s)
Restarted GMRES(30)	1.15 (s)	0.77 (s)	0.45 (s)	0.50 (s)	1.17 (s)
CurlCurl4	**Number of processes**				
Linear solver	**32**	**64**	**128**	**256**	**512**
TR_AAR_LSQR(6,10)	152.07 (s)	93.17 (s)	51.44 (s)	33.64 (s)	49.49 (s)
TR_AAR_NE(6,10)	89.01 (s)	58.73 (s)	30.78 (s)	18.76 (s)	18.12 (s)
Restarted GMRES(10)	120.34 (s)	61.42 (s)	32.62 (s)	21.19 (s)	20.29 (s)
Restarted GMRES(30)	120.33 (s)	61.74 (s)	32.22 (s)	20.73 (s)	20.34 (s)
dielFilterV2real	**Number of processes**				
Linear solver	**32**	**64**	**128**	**256**	**512**
TR_AAR_LSQR(6,10)	-	1,340.90 (s)	975.99 (s)	832.11 (s)	1,386.38 (s)
TR_AAR_NE(6,10)	1,039.92 (s)	721.01 (s)	560.64 (s)	474.52 (s)	545.67 (s)
Restarted GMRES(10)	-	1,233.76 (s)	816.01 (s)	907.48 (s)	925.95 (s)
Restarted GMRES(30)	-	1,219.00 (s)	882.62 (s)	940.95 (s)	929.63 (s)
Transport	**Number of processes**				
Linear solver	**32**	**64**	**128**	**256**	**512**
TR_AAR_LSQR(6,10)	21.25 (s)	12.24 (s)	7.05 (s)	6.94 (s)	14.66 (s)
TR_AAR_NE(6,10)	14.89 (s)	8.20 (s)	4.69 (s)	4.53 (s)	7.44 (s)
Restarted GMRES(10)	8.31 (s)	5.77 (s)	3.37 (s)	3.17 (s)	5.22 (s)
Restarted GMRES(30)	8.30 (s)	5.92 (s)	3.11 (s)	2.89 (s)	5.10 (s)

Table 4.6: `MPI` experiments. AMG preconditioner with a V-cycle made of 5 levels and a coarsening factor equal to 3.

ditioning options explored, some of the problems do not allow the construction of a numerically stable preconditioner or do not allow convergence for the linear solvers tested. This factor limits the impact of the conclusion that can be drawn from the numerical experiments of this chapter. However, the results obtained are promising enough to encourage further studies in this respect.

A significant limitation we encountered is the fact that the matrix has to be imported by reading it from a file. A more efficient approach may be to directly interface the code of the linear solver with the code that takes care of generating the problem. This way, the time consuming writing/reading steps to import the matrix can be circumvented.

Moreover, an object-oriented implementation of the code in C++ may be of interest. In fact, this would allow the user to exploit HPC numerical linear algebra libraries written in C++ such as `Trilinos` [67].

Chapter 5

Monte Carlo Linear Solvers

This chapter essentially coincide with the work published in [8].

5.1 Introduction

The next generation of computational science applications will require numerical solvers that are both reliable and capable of high performance on projected exascale platforms. In order to meet these goals, solvers must be resilient to soft and hard system failures, provide high concurrency on heterogeneous hardware configurations, and retain numerical accuracy and efficiency. In this chapter we focus on the solution of large sparse systems of linear equations, for example of the kind arising from the discretization of partial differential rquations (PDEs). A possible approach is to try to adapt existing solvers (such as preconditioned Krylov subspace or multigrid methods) to the new computational environments, and indeed several efforts are under way in this direction; see, e.g., [1, 26, 37, 50, 58] and references therein. An alternative approach is to investigate new algorithms that can address issues of resiliency, particularly fault tolerance and hard processor failures, naturally. An example is provided by the recently proposed *Monte Carlo Synthetic Acceleration Methods* (MCSA), see [24, 55]. In these methods, an underlying (deterministic) stationary Richardson iterative method is combined with a stochastic, Monte Carlo-based “acceleration” scheme. Ideally, the accelerated scheme will converge to the solution of the linear system in far fewer (outer) iterations than the basic scheme without Monte Carlo acceleration, with the added advantage that most of the computational effort is now relegated to the Monte Carlo portion of the algorithm, which is highly parallel and offers a more straightforward path to resiliency than standard, deterministic solvers. In addition, a careful combination of the Richardson and Monte Carlo parts of the algorithm allows to circumvent the well known problem of slow Monte Carlo error reduction; see [24].

Numerical evidence presented in [24] suggests that MCSA can be competitive, for certain classes of problems, with established deterministic solvers such as preconditioned conjugate gradients and Generalized Minimum Residual (GMRES). So far, however, no theoretical analysis of the convergence properties of these solvers has been carried out. In particular, it is not clear a priori whether the method, applied to a particular linear system, will converge. Indeed, the convergence of the underlying preconditioned Richardson iteration is not sufficient, in general, to guarantee the convergence of the MCSA-accelerated iteration. In other words, it is quite possible that the stochastic “acceleration” part of the algorithm may actually cause the hybrid method to diverge or stagnate.

In this chapter we address this fundamental issue, discussing both necessary and sufficient conditions for convergence. We also discuss the choice of splitting, or preconditioner, and illustrate our findings by means of numerical experiments.

The chapter is organized as follows. In Section 5.2 we provide an overview of existing Monte Carlo linear solver algorithms. In Section 5.3 we will discuss the convergence behavior of stochastic solvers, including a discussion of classes of matrices for which convergence can be guaranteed. Section 5.4 provides some numerical results illustrating properties of the various approaches and in Section 5.5 we give our conclusions.

5.2 Stochastic linear solvers

Linear solvers based on stochastic techniques have a long history, going back to the famed 1928 paper by Courant, Friedrichs, and Lewy on finite difference schemes for PDEs [16]. Many authors have considered linear solvers based on Monte Carlo techniques, with important early contributions by Curtiss [17] and by Forsythe and Leibler [28]. More recent works include [2, 34], and [20], among others. Until recently, these methods have had mixed success at best, due to their generally inferior performance when compared to state-of-the-art deterministic solvers like multigrid or preconditioned Krylov methods. Current interest in resilient solvers, where some performance may be traded off for increased robustness in the presence of faults, has prompted a fresh look at methods incorporating Monte Carlo ideas [24, 55, 56].

As mentioned in [20], Monte Carlo methods may be divided into two broad classes: *direct methods*, such as those described in [20,21], and *iterative methods*, which refer to techniques such as those presented in [33, 34]; see also [24, 56]. The first type consists of purely stochastic schemes, therefore the resulting error with respect to the exact solution is made of just a stochastic component. In contrast, the iterative Monte Carlo methods utilize more traditional iterative algorithms alongside the stochastic approach, generating two types of error: a *stochastic* one and a *systematic* one. In practice, it may be difficult to separate the two components; nevertheless, awareness of this intrinsic structure is useful, as it allows algorithm designers some flexibility in the choice of what part of the algorithm to target for refinement (e.g., trading off convergence speed for resilience by balancing the number of “deterministic” outer iterations against the number of random walks to be used within each iteration).

Consider a nonsingular linear system as in Equation (2.1), which Equation (2.1) can be recast as a fixed point problem as in Equation (2.3). Assuming that the spectral radius $\rho(H) < 1$, the solution to (2.3) can be written in terms of a power series in H (Neumann series):

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

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

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

$$x_i = f_i + \sum_{\ell=1}^{\infty}\sum_{k_1=1}^{n}\sum_{k_2=1}^{n}\cdots\sum_{k_\ell=1}^{n} H_{i,k_1}H_{k_1,k_2}\cdots H_{k_{\ell-1},k_\ell}f_{k_\ell}. \tag{5.1}$$

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

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

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

5.2.1 Forward method

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

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

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

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

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

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

where

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

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

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

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

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

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

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

as the random variable associated with a specific realization ν. We can thus define the estimator θ as

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

where ν ranges over all possible realizations. P_ν is the probability associated with a specific realization of the random walk. It can be proved (see [33] and [34]) that

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

and

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

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

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

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

$$\theta = x_i = f_i + \sum_{l=1}^{\infty} \sum_{k_1=1}^{n} \sum_{k_2=1}^{n} \cdots \sum_{k_\ell=1}^{n} P_{i,k_1} P_{k_1,k_2} \cdots P_{k_{\ell-1},k_\ell} w_{i,k_1} w_{k_1,k_2} \cdots w_{k_{\ell-1},k_\ell} f_{k_\ell}. \quad (5.2)$$

As regards the variance, we recall the following relation:

$$Var\Bigg[\sum_{\ell=0}^{\infty} W_\ell f_{k_\ell}\Bigg] = E\Bigg[\sum_{\ell=0}^{\infty} W_\ell^2 f_{k_\ell}^2\Bigg] - \Bigg(E\Bigg[\sum_{\ell=0}^{\infty} W_\ell f_{k_\ell}\Bigg]\Bigg)^2. \quad (5.3)$$

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

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

$$E\Bigg[\sum_{\ell=0}^{\infty} W_\ell f_{k_\ell}\Bigg] < \infty, \quad (5.4)$$

$$E\Bigg[\sum_{\ell=0}^{\infty} W_\ell^2 f_{k_\ell}^2\Bigg] < \infty. \quad (5.5)$$

The forward method presented above, however, has the limitation of employing an entire set of realizations to estimate just a single entry of the solution at a time. Hence, in order to estimate the entire solution vector for Eq. (2.1), we have to employ a separate set of realizations for each entry in the solution vector. This limitation can be circumvented by the adjoint method which we describe below.

Remark 5. It is important to note that in order to construct the random walks, access to the individual entries of H is required. Hence, H needs to be formed explicitly and therefore must be sparse in order to have a practical algorithm.

5.2.2 Adjoint method

A second Monte Carlo method can be derived by considering the linear system adjoint to (2.1):

$$A^T \mathbf{y} = \mathbf{d}, \tag{5.6}$$

where $\mathbf{y}$ and $\mathbf{d}$ are the adjoint solution and source term. Equation (5.6) can be recast in a manner similar to (2.4):

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

Note that $\rho(H^T) = \rho(H) < 1$, hence convergence of the Neumann series (fixed point iteration) for (2.1) guarantees convergence for the adjoint system (5.6).

Exploiting the following inner product equivalence:

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

it follows that

$$\langle \mathbf{x}, \mathbf{d} \rangle = \langle \mathbf{y}, \mathbf{f} \rangle \,. \tag{5.7}$$

By writing the Neumann series for the solution to (5.6) we have:

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

and focusing on a single entry of the solution vector:

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

The undetermined quantities in the dual problem (5.6) are $\mathbf{y}$ and $\mathbf{d}$. Therefore, two constraints are required: the first constraint is Eq. (5.7) and as a second constraint we select $\mathbf{d}$ to be one of the standard basis vectors. Applying this choice of $\mathbf{d}$ to (5.7) we get:

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

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

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

and the initial weight:

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

The transition probability is defined as

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

and the sequence of weights as follows:

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

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

$$\begin{aligned}\theta_i &= E\left[\sum_{\ell=0}^{\infty} W_\ell \delta_{k_\ell,i}\right] \\ &= \sum_{\ell=0}^{\infty}\sum_{k_0=1}^{n}\sum_{k_1=1}^{n}\sum_{k_2=1}^{n}\cdots\sum_{k_\ell=1}^{n} f_{k_0} P_{k_0,k_1} P_{k_1,k_2}\cdots P_{k_{\ell-1},K_\ell} w_{k_0,k_1}\cdots w_{k_{\ell-1},k_\ell}\delta_{k_\ell,i}.\end{aligned} \tag{5.8}$$

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

The forward method adds a contribution to the component of the solution vector where the random walk began, based on the value of the source vector in the state in which the walk currently resides. The adjoint method, on the other hand, adds a contribution to the component of the solution vector where the random walk currently resides based on the value of the source vector in the state in which the walk began. The Kronecker delta at the end of the series (5.8) represents a filter, indicating that only a subset of realizations contribute to the jth component of the solution vector.

The variance is given by

$$Var\left[\sum_{\ell=0}^{\infty} W_\ell \delta_{k_\ell,i}\right] = E\left[\sum_{\ell=0}^{\infty} W_\ell^2 \delta_{k_\ell,i}\right] - \left(E\left[\sum_{\ell=0}^{\infty} W_\ell \delta_{k_\ell,i}\right]\right)^2, \qquad i = 1,\dots,n. \tag{5.9}$$

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

$$E\left[\sum_{\ell=0}^{\infty} W_\ell \delta_{k_\ell,i}\right] < \infty \qquad i = 1,\dots,n \tag{5.10}$$

and

$$E\left[\sum_{\ell=0}^{\infty} W_\ell^2 \delta_{k_\ell,i}\right] < \infty, \qquad i = 1,\dots,n. \tag{5.11}$$

The main advantage of this method, compared to the forward one, consists in the fact that a single set of realizations is used to estimate the entire solution vector. Unless only a small portion of the problem is of interest, this property often leads to the adjoint method being favored over the forward method. In other terms, the adjoint method should be preferred when approximating the solution globally over the entire computational domain, while the forward method is especially useful when approximating the solution locally.

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

$$\begin{aligned} \theta_i &= E\Big[f_i + \textstyle\sum_{\ell=0}^{\infty} W_\ell H^T_{k_\ell,i}\Big] \\ &= f_i + \sum_{\ell=0}^{\infty}\sum_{k_0=1}^{n}\sum_{k_1=1}^{n}\sum_{k_2=1}^{n}\cdots\sum_{k_\ell=1}^{n} f_{k_0} P_{k_0,k_1} P_{k_1,k_2}\cdots P_{k_{\ell-1},k_\ell} w_{k_0,k_1}\cdots w_{k_{\ell-1},k_\ell} H^T_{k_\ell,i}. \end{aligned} \tag{5.12}$$

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

$$Var\Bigg[\sum_{\ell=0}^{\infty} W_\ell H^T_{k_\ell,i}\Bigg] = E\Bigg[\sum_{\ell=0}^{\infty} W_\ell^2 H^T_{k_\ell,i}\Bigg] - \Bigg(E\Bigg[\sum_{\ell=0}^{\infty} W_\ell H^T_{k_\ell,i}\Bigg]\Bigg)^2, \qquad i = 1,\dots,n. \tag{5.13}$$

5.2.3 Hybrid stochastic/deterministic methods

The direct methods described in Sections 5.2.1 and 5.2.2 suffer from a slow rate of convergence due to the $\frac{1}{\sqrt{N}}$ behavior dictated by the central limit theorem (N here is the number of random walks used to estimate the solution). Furthermore, when the spectral radius of the iteration matrix is close to unity, each individual random walk may require a large number of transitions to approximate the corresponding components in the Neumann series. To offset the slow convergence of the central limit theorem, schemes have been proposed which combine traditional fixed point iterative methods with the stochastic solvers. The first such method, due to Halton, was termed the Sequential Monte Carlo (SMC) method, and can be written as follows.

Algorithm 8: Sequential Monte Carlo

```
**Data:** $H$, $\mathbf{b}$, $\mathbf{x}_0$
**Result:** $\mathbf{x}_{num}$
1 $l = 0$;
2 while not reached convergence do
3     $\mathbf{r}^l = \mathbf{b} - A\mathbf{x}^l$;
4     $\delta\mathbf{x}^{l+1} \approx (I - H)^{-1}\mathbf{r}^l$;        % Computed via Standard MC;
5     $\mathbf{x}^{l+1} = \mathbf{x}^l + \delta\mathbf{x}^{l+1}$;
6     $l = l + 1$;
7 end
8 $\mathbf{x}_{num} = \mathbf{x}^{l+1}$;
```

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

$$(I - H)\delta\mathbf{x}^{l+1} = \mathbf{r}^l. \tag{5.14}$$

If this equation were to be solved exactly, the corresponding approximation $\mathbf{x}^{l+1} = \mathbf{x}^l + \delta\mathbf{x}^{l+1}$ would be the exact solution to the linear system. This is of course impractical, since solving (5.14) is equivalent to solving the original linear system (2.1). Instead, the correction is obtained by solving (5.14) only approximately, using a Monte Carlo method. Because Monte Carlo is only applied within a single iteration, the central limit theorem is only applicable within that iteration rather than to the overall convergence behavior of the algorithm. This allows a trade-off between the amount of time and effort spent on the inner (stochastic) and outer (deterministic) iterations, which can take into account the competing goals of reliability and rapid convergence.

A further extension of Halton's method, termed *Monte Carlo Synthetic Acceleration* (MCSA), has been recently introduced in [56] and [24]. The MCSA algorithm can be written as:

Algorithm 9: Monte Carlo Synthetic Acceleration

```
   Data: $H$, $\mathbf{b}$, $\mathbf{x}_0$
   Result: $\mathbf{x}_{num}$
1  $l = 0$;
2  while not reached convergence do
3      $\mathbf{x}^{l+\frac{1}{2}} = H\mathbf{x}^l + \mathbf{b}$;
4      $\mathbf{r}^{l+\frac{1}{2}} = \mathbf{b} - A\mathbf{x}^{l+\frac{1}{2}}$;
5      $\delta\mathbf{x}^{l+\frac{1}{2}} \approx (I - H)^{-1}\mathbf{r}^{l+\frac{1}{2}}$;        % Computed via Standard MC;
6      $\mathbf{x}^{l+1} = \mathbf{x}^{l+\frac{1}{2}} + \delta\mathbf{x}^{l+\frac{1}{2}}$;
7      $l = l + 1$;
8  end
9  $\mathbf{x}_{num} = \mathbf{x}^{l+1}$;
```

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

Obviously, a minimum requirement is that the linear system can be written in the form (2.4) with $\rho(H) < 1$. This is typically achieved by preconditioning. That is, we find an invertible matrix P such that $H = I - P^{-1}A$ satisfies $\rho(H) < 1$, and we apply the method to the fixed point problem (2.4) where $H = I - P^{-1}A$ and $\mathbf{f} = P^{-1}\mathbf{b}$. In other words, the underlying deterministic iteration is a preconditioned Richardson iteration. Various choices of the preconditioner are possible; a detailed discussion of this issue is deferred until Section 5.3.5. Here we note only that because we need explicit knowledge of the entries of H, not all preconditioning choices are viable; in particular, P needs to be such that $H = I - P^{-1}A$ retains a high degree of sparsity. Unless otherwise specified, below we assume that the transformation of the original linear system (2.1) to the fixed point form (2.4) with $\rho(H) < 1$ has already been carried out.

5.3 Convergence behavior of stochastic methods

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

$$\rho(H) < 1, \tag{5.15}$$

where H is the iteration matrix of the fixed point scheme. Indeed, we can see from (5.2) and (5.8) that the expected value is expressed in terms of power series of H, and the condition $\rho(H) < 1$ is a necessary and sufficient condition for the Neumann series to converge.

Next, we address the finiteness requirement for the second moment. Equations (5.3) and (5.9) for the forward and the adjoint method, respectively, show that the second moment can be reinterpreted as a power series with respect to the matrices defined as follows:

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

and

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

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

$$\rho(\hat{H}) < 1. \tag{5.16}$$

Hence, condition (5.15) is required for a generic fixed point scheme to reach convergence, whereas the extra condition (5.16) is typical of the stochastic schemes studied in this work. Moreover, since the finiteness of the variance automatically entails the finiteness of the expected value, we can state that (5.16) implicitly entails (5.15), whereas the converse is not true in general.

5.3.1 Necessary and sufficient conditions

Here we report some results presented in [57] and [35], concerning necessary conditions and sufficient conditions for convergence. In particular, these papers discuss suitable choices for constructing the transition probability matrix, P.

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

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

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

$$\begin{aligned} &\text{Forward Method:} \quad H_{ij} \neq 0 \Rightarrow P_{ij} \neq 0, \\ &\text{Adjoint Method:} \quad H_{ji} \neq 0 \Rightarrow P_{ij} \neq 0. \end{aligned}$$

The following auxiliary result can be found in [35].

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

(i) for any probability distribution vector $\boldsymbol{\beta} = (\beta_1, \beta_2, \dots, \beta_N)^T$ satisfying the transition conditions, $\displaystyle\sum_{k=1}^{N} \frac{g_k^2}{\beta_k} \geq \left(\sum_{k=1}^{N} |g_k| \right)^2$; moreover, the lower bound is attained for the probability vector defined by $\beta_k = \dfrac{|g_k|}{\sum_{k=1}^{N}|g_k|}$;

(ii) there always exists a probability vector $\boldsymbol{\beta}$ such that $\displaystyle\sum_{k=1}^{N}\frac{g_k^2}{\beta_k} \geq c$, for all $c > 1$.

Consider now a generic realization of a random walk, truncated at a certain kth step:

$$\nu_k : \; r_0 \to r_1 \to r_2 \to \cdots \to r_k,$$

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

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

Then, the following result presented in [35] holds.

Theorem 5.3.1. *(Forward method version) Let $H \in \mathbb{R}^{n\times n}$ be such that $\|H\|_\infty < 1$. Consider ν_k as the realization of a random walk ν truncated at the kth step. Then, there always exists a transition matrix P such that $Var\Big(X(\nu_k)\Big) \to 0$ and $Var\Big(\sum_\nu X(\nu_k)\Big)$ is bounded as $k \to \infty$.*

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

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

then we can state a theorem analogous to 5.3.1.

Theorem 5.3.2. *(Adjoint method version) Let $H \in \mathbb{R}^{n\times n}$ with $\|H\|_1 < 1$. Consider ν_k as the realization of a random walk ν truncated at the kth step. Then, there always exists a transition matrix P such that $Var\Big(X(\nu_k)\Big) \to 0$ and $Var\Big(\sum_\nu X(\nu_k)\Big)$ is bounded as $k \to \infty$.*

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

The connection between Lemma 1 and Theorems 5.3.1-5.3.2 will be explained in the next section, dedicated to the definition of transition probabilities.

5.3.2 Construction of transition probabilities

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

Uniform probabilities

With this approach, the transition matrix P is such that all the transitions corresponding to each row have equal probability of occurring:

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

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

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

Weighted probabilities

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

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

and

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

where $p \in \mathbb{N}$. The case $p = 1$ is called *Monte Carlo Almost Optimal* (MAO). The reason for the "almost optimal" designation can be understood looking at Lemma 1, as the quantity $\sum_{k=1}^{N} \frac{g_k^2}{\beta_k}$ is minimized when the probability vector is defined as $\beta_k = \frac{|g_k|}{\sum_{k=1}^{N} |g_k|}$. Indeed, Lemma 1 implies that the almost optimal probability minimizes the ∞-norm of $\hat{H}$ for the forward method and the 1-norm of $\hat{H}$ for the adjoint method, since the role of $\mathbf{g}$ in Lemma 1 is played by the rows of H in the former case and by the columns of H in the latter one. This observation provides us with easily computable upper bounds for $\rho(\hat{H})$.

5.3.3 Classes of matrices with guaranteed convergence

On the one hand, sufficient conditions for convergence of Monte Carlo linear solvers are very restrictive; see, e.g., [57] and [35]. On the other hand, the necessary and sufficient condition in [35] requires knowledge of $\rho(\hat{H})$, which is not readily available. Note that explicit computation of $\rho(\hat{H})$ is quite expensive, comparable to the cost of solving the original linear system. While ensuring that $\rho(H) < 1$ (by means of appropriate preconditioning) is in many cases possible, guaranteeing that $\rho(\hat{H}) < 1$ is generally much more problematic.

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

SDD matrices

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

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

$$|a_{ij}| > \sum_{\substack{i=1\\ i\neq j}}^{n} |a_{ij}|. \tag{5.17}$$

Definition 2. A matrix $A \in \mathbb{R}^{n\times n}$ is strictly diagonally dominant by columns if A^T is strictly diagonally dominant by rows, i.e.,

$$|a_{ij}| > \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}|. \tag{5.18}$$

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

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

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

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

Consequently,

$$\sum_{j=1}^{n}|\hat{H}_{ij}| = \sum_{j=1}^{n}\hat{H}_{ij} = \Big(\sum_{j=1}^{n}|H_{ij}|\Big)\Big(\sum_{k=1}^{n}|H_{ik}|\Big) = \Big(\sum_{j=1}^{n}|H_{ij}|\Big)^2 < 1, \qquad \forall i = 1, \cdots, n.$$

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

On the other hand, if (5.18) holds, we can apply right diagonal (Jacobi) preconditioning, which results in an iteration matrix $H = I - AP^{-1}$ such that $\|H\|_1 < 1$. In this case, by a similar reasoning we conclude that the adjoint method converges, owing to $\|\hat{H}\|_1 < 1$; however, nothing can be said *a priori* about the forward method.

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

GDD matrices

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

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

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

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

A proper subclass of the class of GDD matrices is represented by the nonsingular M-matrices. Recall that A is a nonsingular M-matrix if it is of the form $A = rI - B$ where B is nonnegative and $r > \rho(B)$. It can be shown (see, e.g., [11]) that a matrix $A \in \mathbb{R}^{n\times n}$ is a nonsingular M-matrix if and only if there exists a positive diagonal matrix Δ such that $A\Delta$ is SDD by rows. Clearly, every nonsingular M-matrix is GDD.

It is well known (see, e.g., [5]) that the classical Jacobi, Block Jacobi and Gauss–Seidel splittings are convergent if A is a nonsingular M-matrix. However, this is not enough to ensure the convergence of MC schemes based on the corresponding fixed point (preconditioned Richardson) iteration, since in general we cannot expect that $\rho(\hat{H}) < 1$.

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

In [40], the author presents a procedure to determine whether a given matrix $A \in \mathbb{C}^{n\times n}$ is GDD (in which case the diagonal scaling that makes A SDD is produced), or not. The algorithm can be described as follows.

This procedure, which in practice converges very fast, turns a generalized diagonally dominant matrix (in particular, a nonsingular M-matrix) into a strictly diagonally dominant matrix by rows. By replacing $S_i = \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}|$ at step 1 with $S_j = \sum_{\substack{i=1\\ i\neq j}}^{n} |a_{ij}|$ and by replacing $a_{ji} = a_{ji} \cdot d_i$ with $a_{ji} = a_{ji} \cdot d_j$, we obtain the algorithm that turns a GDD matrix into a matrix that is SDD by columns.

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

Block diagonally dominant matrices

In this section we analyze situations in which block diagonal preconditioning can produce a convergent Monte Carlo linear solver. Assume that A has been partitioned into $p \times p$ block form, and that each diagonal block has size n_i with $n_1 + \cdots + n_p = n$. Assume further that each diagonal block A_{ii} is nonsingular. The iteration matrix

Algorithm 10: Algorithm to determine whether a matrix is GDD

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

$H \in \mathbb{R}^{n \times n}$ resulting from a block diagonal left preconditioning is

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

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

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

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

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

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

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

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

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

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

A sufficient condition for this to happen is that

$$\sum_{\substack{k=1\\k\neq h}}^{p}\|A_{hh}^{-1}A_{hk}\|_\infty < 1, \qquad \forall h = 1, \ldots, p. \tag{5.19}$$

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

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

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

$$\|\tilde{H}\|_\infty < 1 \Rightarrow \|\hat{H}\|_\infty < 1. \tag{5.20}$$

Note that (5.19) also implies that $\|H\|_\infty < 1$.

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

$$(\hat{H})^T_{ij} = |H^T_{ij}|\bigg(\sum_{k=1}^{n}|H^T_{ik}|\bigg) = |H_{ji}|\bigg(\sum_{k=1}^{n}|H_{ki}|\bigg).$$

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

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

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

$$\sum_{\substack{h=1 \\ h\neq k}}^{p} \|A_{hh}^{-1}A_{hk}\|_1 < 1, \qquad \forall k = 1, \dots, p. \tag{5.21}$$

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

We say that A is *strictly block diagonally dominant by rows (columns) with respect to a given block partition* if condition (5.19) (respectively, (5.21)) is satisfied relative to that particular block partition. Note that a matrix may be strictly block diagonally dominant with respect to one partition and not to another. We note that these definitions of block diagonal dominance are different from those found in, e.g., [27], and they are easier to check in practice since they do not require computing the 2-norm of the blocks of H.

5.3.4 Adaptive methods

In formulas (5.2) and (5.8), the estimation of the solution to the linear system (2.1) involves infinite sums, which in actual computation have to be truncated. In the following we discuss criteria to decide the number of steps to be taken in a single random walk as well as the number of random walks that need to be performed at each Richardson iteration.

History length

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

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

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

and

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

are good approximations of (5.2) and (5.8), respectively.

In [55] a criterion was given which is applicable to both forward and adjoint methods. It requires to set up a relative weight cutoff threshold W_c and to look for a step m such that

$$W_m \leq W_c W_0\,. \tag{5.22}$$

In (5.22), W_0 is the value of the weight at the initial step of the random walk and W_m is the value of the weight after m steps. We will adopt a similar strategy in this work.

Number of random walks

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

The expression for the variance of the forward method is given by formula (5.3). In this context, a reasonable criterion to determine the number $\tilde{N}_i$ of random walks to be run is to set a threshold ε_1 and determine $\tilde{N}_i$ such that

$$\frac{\sqrt{Var[\theta_i]}}{|E[\theta_i]|} < \varepsilon_1, \quad i = 1, \dots, n. \tag{5.23}$$

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

As concerns the adjoint method, the estimation of the variance is given in formula (5.9). A possible criterion for the adaptive selection of the number $\tilde{N}$ of random walk, in this situation, is that it satisfies the condition

$$\frac{\|\boldsymbol{\sigma}^{\tilde{N}}\|_1}{\|\mathbf{x}\|_1} < \varepsilon_1, \tag{5.24}$$

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

The criteria introduced above can be exploited to build an a posteriori adaptive algorithm, capable of identifying the minimal value of $\tilde{N}$ that verifies (5.23) or (5.24), respectively. Algorithms 11 and 12 describe the Monte Carlo approaches with the adaptive criteria.

The use of the adaptive approach for the selection of the number of histories has

Algorithm 11: A posteriori adaptive Forward Monte Carlo

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

Algorithm 12: A posteriori adaptive Adjoint Monte Carlo

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

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

$\rho(H)$ is close to 1, since in this case the Richardson step is less effective in dampening the uncertainty coming from the previous iterations.

5.3.5 Preconditioning

As noted at the end of Section 5.2, left preconditioning can be incorporated into any Monte Carlo linear solver algorithm by simply substituting A with $P^{-1}A$ and $\mathbf{b}$ with $P^{-1}\mathbf{b}$ in (2.1); i.e., we set $H = I - P^{-1}A$ and $\mathbf{f} = P^{-1}\mathbf{b}$ in (2.4). Right preconditioning can also be incorporated by rewriting (2.1) as $AP^{-1}\mathbf{y} = \mathbf{b}$, with $\mathbf{y} = P\mathbf{x}$; i.e., we set $H = I - AP^{-1}$ and replace $\mathbf{x}$ by $\mathbf{y}$ in (2.4). The solution $\mathbf{x}$ to the original system (2.1) is then given by $\mathbf{x} = P^{-1}\mathbf{y}$. Likewise, split (left and right) preconditioning can also be used. The Monte Carlo process, however, imposes some constraints on the choice of preconditioner. Most significantly, because the transition probabilities are built based on the values of the iteration matrix H, it is necessary to have access to the entries of the preconditioned matrix $P^{-1}A$ (or AP^{-1}). Therefore, we are limited to preconditioners that enable explicitly forming the preconditioned matrix while retaining some of the sparsity of the original matrix.

One possible preconditioning approach involves either diagonal or block diagonal preconditioning (with blocks of small or moderate size). Diagonal preconditioning does not alter the sparsity of the original coefficient matrix, whereas block diagonal preconditioning will incur a moderate amount of fill-in in the preconditioned matrix if the blocks are not too large. From the discussions in Section 5.3.3 and 18, selecting a diagonal or block diagonal preconditioner guarantees convergence of the Monte Carlo schemes for matrices that are strictly diagonally dominant or block diagonally dominant, respectively. In addition, M-matrices that are not strictly or block diagonally dominant can also be dealt with by first rescaling A so that it becomes strictly diagonally dominant, as discussed in Section 5.3.3.

In principle, other standard preconditioning approaches can also be used in an attempt to achieve both $\rho(H) < 1$ and $\rho(\hat{H}) < 1$ while still retaining sparsity in the preconditioned matrix. One possibility is the use of incomplete LU factorizations [7, 52]. If $P = LU$ is the preconditioner with sparse triangular factors L and U, then $P^{-1}A$ can in principle be formed explicitly provided that the sparsity in L, U and A is carefully exploited in the forward and back substitutions needed to form $(LU)^{-1} = U^{-1}(L^{-1}A)$. In general, however, this results in a rather full preconditioned matrix; sparsity needs to be preserved by dropping small entries in the resulting matrix.

Another class of preconditioners that are potentially of interest for use with Monte Carlo linear solvers are approximate inverse preconditioners [7]. In these algorithms, an approximation to the inverse of A is generated directly and the computation of the preconditioned matrix reduces to one or more sparse matrix-matrix products, a relatively straightforward task. As with ILU factorizations, multiple versions of approximate inverse preconditioning exist which may have different behavior in terms of effectiveness of the preconditioner versus the resulting reduction in sparsity.

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

5.3.6 Considerations about computational complexity

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

Recalling formula (5.2) for the entry-wise estimate for the solution with the forward method, the cost of reconstructing the entire solution vector is

$$\text{Forward:} \quad \mathcal{O}(N \cdot k_\ell \cdot n), \tag{5.25}$$

where N is the number of histories, k_ℓ is the length of each random walk, and n is the number of unknowns. As concerns the adjoint method, formula (5.8) leads to

$$\text{Adjoint:} \quad \mathcal{O}(N \cdot k_\ell). \tag{5.26}$$

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

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

$$\text{Sequential MC with forward MC update:} \quad \mathcal{O}((n+1) \cdot N \cdot k_\ell) \tag{5.27}$$

using the forward MC as an inner solver, and

$$\text{Sequential MC with adjoint MC update:} \quad \mathcal{O}(n + N \cdot k_\ell) \tag{5.28}$$

using the adjoint MC as an inner solver.

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

$$\text{MCSA with forward MC update:} \quad \mathcal{O}((2n+1) \cdot N \cdot k_\ell), \tag{5.29}$$

whereas using the adjoint MC we find

$$\text{MCSA with adjoint MC update:} \quad \mathcal{O}(2n + N \cdot k_\ell). \tag{5.30}$$

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

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

Table 5.1: Properties of the matrices employed for the checks on adaptivity.

5.4 Numerical results

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

5.4.1 Numerical tests with adaptive methods

In this subsection we study experimentally the adaptive approaches discussed in Section 5.3.4. For this purpose, we restrict our attention to standard Monte Carlo linear solvers.

For these tests we limit ourselves to small matrices, primarily because the numerical experiments are being computed on a standard laptop and the computational cost of Monte Carlo methods rapidly becomes prohibitive on such machines. The smallest of these matrices represents a finite difference discretization of a 1D reaction-diffusion equation, the second one is a discrete 2D Laplacian with zero Dirichlet boundary conditions, the third a steady 2D advection-diffusion operator discretized by quadrilateral linear finite elements using the IFISS package [22], and the fourth one results from a finite volume discretization of a thermal radiation diffusion equation (Marshak problem). The first and the last of these matrices are strictly diagonally dominant by both rows and columns. For all the problems, left diagonal preconditioning is applied. Details about these matrices are given in Table 5.1.

We present results for both the Forward and the Adjoint Monte Carlo methods.

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

Table 5.2: Forward Monte Carlo. Adaptive selection of histories, $\varepsilon_1 = 0.1$.

As concerns the Forward method, we set the maximum number of histories per entry to 10^7. In Tables 5.2 and 5.3 we show results for values of the adaptive threshold (5.23) equal to $\varepsilon_1 = 0.1$ and $\varepsilon_1 = 0.01$, respectively. A batch size of two is used at each adaptive check to verify the magnitude of the apparent relative standard deviation. As expected, results are aligned with the convergence rate predicted by the Central Limit Theorem. Indeed, decreasing by a factor of 10 the tolerance ε_1 we see that the relative error undergoes a decrease of the same order, requiring roughly one hundred times more histories. In the case of the 2D Laplacian we actually have an increase of a factor close to 400 in the number of histories, but the relative error is decreased by more than thirty times. For this particular example, the forward method overestimates the number of histories needed to satisfy a prescribed reduction on the standard deviation.

As regards the Adjoint Monte Carlo, at each adaptive check the number of random walks employed is increased by ten. A maximum number of histories equal to 10^{10} is set. Tables 5.4, 5.5 and 5.6 show results for the different test cases using adaptive thresholds (5.24) with values $\varepsilon_1 = 0.1$, $\varepsilon_1 = 0.01$ and $\varepsilon_1 = 0.001$, respectively. By comparing the reported errors, it is clear that a decrease in the value of the threshold induces a reduction of the relative error of the same order of magnitude. This confirms the effectiveness of the adaptive selection of histories with an error reduction goal. Each decrease in the error by an order of magnitude requires an increase in the total number of histories employed by two orders of magnitude, as expected.

The same simulations can be run resorting to the expected value estimator. Re-

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

Table 5.3: Forward Monte Carlo. Adaptive selection of histories, $\varepsilon_1 = 0.01$.

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

Table 5.4: Collision estimator – Adjoint Monte Carlo. Adaptive selection of histories, $\varepsilon_1 = 0.1$.

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

Table 5.5: Collision estimator – Adjoint Monte Carlo. Adaptive selection of histories, $\varepsilon_1 = 0.01$.

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

Table 5.6: Collision estimator – Adjoint Monte Carlo. Adaptive selection of histories, $\varepsilon_1 = 0.001$.

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

Table 5.7: Expected value estimator – Adjoint Monte Carlo. Adaptive selection of histories, $\varepsilon_1 = 0.1$

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

Table 5.8: Expected value estimator – Adjoint Monte Carlo. Adaptive selection of histories, $\varepsilon_1 = 0.01$.

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

Table 5.9: Expected value estimator – Adjoint Monte Carlo. Adaptive selection of histories, $\varepsilon_1 = 0.001$.

sults are shown in the Tables 5.7, 5.8 and 5.9 for the threshold values of $\varepsilon_1 = 0.1$, $\varepsilon_1 = 0.01$ and $\varepsilon_1 = 0.001$ respectively. As it can be noticed, in terms of error scaling the results are quite similar to the ones obtained with the collision estimator. As regards the number of histories needed to reach a prescribed accuracy, the orders of magnitude are the same for both the collision and the expected value estimators. However, the expected value estimator requires in most cases a smaller number of realizations. This behavior becomes more pronounced as the value of the threshold decreases, making the computation increasingly cost-effective.

5.4.2 Preconditioning approaches

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

The values of the two spectral radii with diagonal preconditioning have already been shown in Table 5.4. Here we consider the effect of block diagonal preconditioning

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

Table 5.10: Behavior of $\rho(H)$ and $\rho(\hat{H})$ for the 2D Laplacian with block diagonal preconditioning.

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

Table 5.11: Behavior of $\rho(H)$ and $\rho(\hat{H})$ for the 2D advection-diffusion problem with block diagonal preconditioning.

for different block sizes, and the use of the factorized sparse approximate inverse preconditioner AINV [9, 10] for different values of the drop tolerance (which controls the sparsity in the approximate inverse factors). Intuitively, with these two types of preconditioners both $\rho(H)$ and $\rho(\hat{H})$ should approach zero for increasing block size and decreasing drop tolerance, respectively; however, the convergence need not be monotonic in general, particularly for $\rho(\hat{H})$. This somewhat counterintuitive behavior is shown in Tables 5.10 and 5.11, where an increase in the size of the blocks used for the block diagonal preconditioner results in an increase of $\rho(\hat{H})$ for both test problems. Note also the very slow rate of decrease in $\rho(H)$ for increasing block size, which is more than offset by the rapid increase in the density of H, which of course implies much higher costs. We mention that for a block size of 30 the 2D discrete Laplacian is block diagonally dominant, but not for smaller block sizes. The 2D discrete advection-diffusion operator is not block diagonally dominant for any of the three reported block sizes.

In Tables 5.12 and 5.13 the values of the spectral radii are shown for the AINV preconditioner with two different values of the drop tolerance [9, 10]. It is interesting to point out that, for the two-dimensional Laplacian, a drop tolerance $\tau = 0.05$

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

Table 5.12: Behavior of $\rho(H)$ and $\rho(\hat{H})$ for the 2D Laplacian with AINV preconditioning.

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

Table 5.13: Behavior of $\rho(H)$ and $\rho(\hat{H})$ for the 2D advection-diffusion problem with AINV preconditioning.

entails $\rho(H) < 1$ but the same does not hold for $\hat{H}$. Both convergence conditions are satisfied by reducing the drop tolerance, but at the price of very high fill-in in the preconditioned matrix.

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

5.4.3 Hybrid methods

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

Poisson problem

Consider the standard 2D Poisson model problem:

$$\begin{cases} -\Delta u = f \quad \text{in} \quad \Omega, \\ u = 0 \quad \text{on} \quad \partial\Omega, \end{cases} \tag{5.31}$$

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

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

We discretize problem (5.31) using standard 5-point finite differences. For $N = 32$ nodes in each direction, we obtain the 900×900 linear system already used in the previous section.

Consider the iteration matrix H corresponding to (left) diagonal preconditioning. It is well known that $\rho(H) < 1$, and indeed we can see from Table 5.1 that $\rho(H) \approx 0.9949$. It is also easy to see that $\|H\|_1 = 1$. In order for the Adjoint Monte Carlo method to converge, it is necessary to have $\rho(\hat{H}) < 1$, too. If an almost optimal probability is used, the Adjoint Monte Carlo method leads to a $\hat{H}$ matrix such that

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

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

- four entries equal to $\frac{1}{4}$ on the rows associated with a node which is not adjacent to the boundary;

- two entries equal to $\frac{1}{4}$ on the rows associated with nodes adjacent to the corner of the square domain;

- three entries equal to $\frac{1}{4}$ on the rows associated with nodes adjacent to the boundary on an edge of the square domain.

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

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

where $\tilde{D}$ is a diagonal matrix with diagonal entries equal to 1, $\frac{1}{2}$ or $\frac{3}{4}$. In particular, $\tilde{d}_i = \text{diag}(\tilde{D})_i = 1$ if the ith row of the discretized Laplacian is associated with a node which is not adjacent to the boundary, $\tilde{d}_i = \text{diag}(\tilde{D})_i = \frac{1}{2}$ if the row is associated with a node of the grid adjacent to the corner of the boundary, and $\tilde{d}_i = \text{diag}(\tilde{D})_i = \frac{3}{4}$ if the associated node of the grid is adjacent to the boundary of the domain far from a corner. Since H is an irreducible nonnegative matrix and $\tilde{D}$ is a positive definite diagonal matrix, we can invoke a result in [29] to conclude that

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

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

Results of numerical experiments are reported in Table 5.14. We compare the purely deterministic Richardson iteration (Jacobi's method in this case) with two hybrid methods, Halton's sequential Monte Carlo and MCSA using the Adjoint method.

The threshold for the adaptive selection of the random walks is set to $\varepsilon_1 = 0.1$. Convergence is attained when the initial residual has been reduced by at least eight orders of magnitude, starting from a zero initial guess. Note that, as expected, the diagonally preconditioned Richardson iteration converges very slowly, since the spectral radius of H is very close to 1. Halton's Sequential Monte Carlo performs quite well on this problem, but even better results are obtained by the Adjoint MCSA method, which converges in one iteration less than Sequential Monte Carlo while requiring far less Monte Carlo histories per iteration. This might be explained by the fact that within each outer iteration, the first relaxation step in the MCSA algorithm refines the accuracy of the solution coming from the previous iteration, before using it as the input for the MC solver at the current iterate. In particular, its effect is to dampen the statistical noise produced by the Monte Carlo linear solver in the estimation of the update $\delta\mathbf{x}^{l+\frac{1}{2}}$ performed at the previous iteration. Therefore, the refinement, or smoothing, accomplished by the Richardson relaxation decreases the number of random walks needed for a prescribed accuracy. This hypothesis is validated by the fact that at the first iteration both Sequential Monte Carlo and MCSA use the same number of histories. Their behaviors start differing from the second iteration on, when the statistical noise is introduced in the estimation of the correction to the current iterate; see Figures 5.1 and 5.2. It can be seen that in all the numerical tests presented in this section there is a sharp increase (from $O(10^3)$–$O(10^5)$ to $O(10^6)$–$O(10^7)$) in the number of histories when going from the first to subsequent iterations. This is due to the introduction of statistical noise coming from the Monte Carlo updating of the solution. Indeed, the stochastic noise, introduced from the second iteration on, increases the uncertainty associated with the estimate of the solution. Therefore, the adaptive criterion forces the algorithm to perform a higher number of histories to achieve a prescribed accuracy.

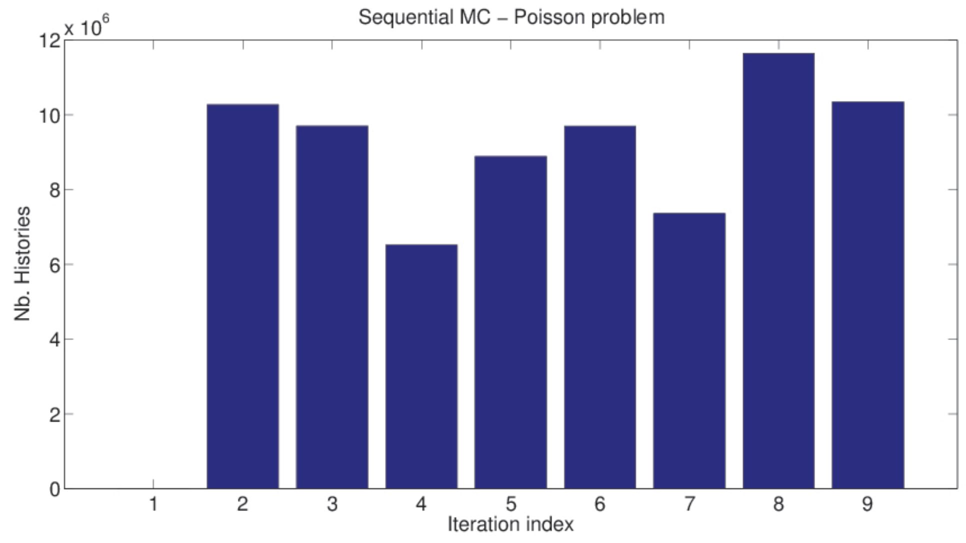

Figure 5.1: Sequential MC - Poisson problem. Number of random walks employed at each iteration. The number for the first iteration is 2000.

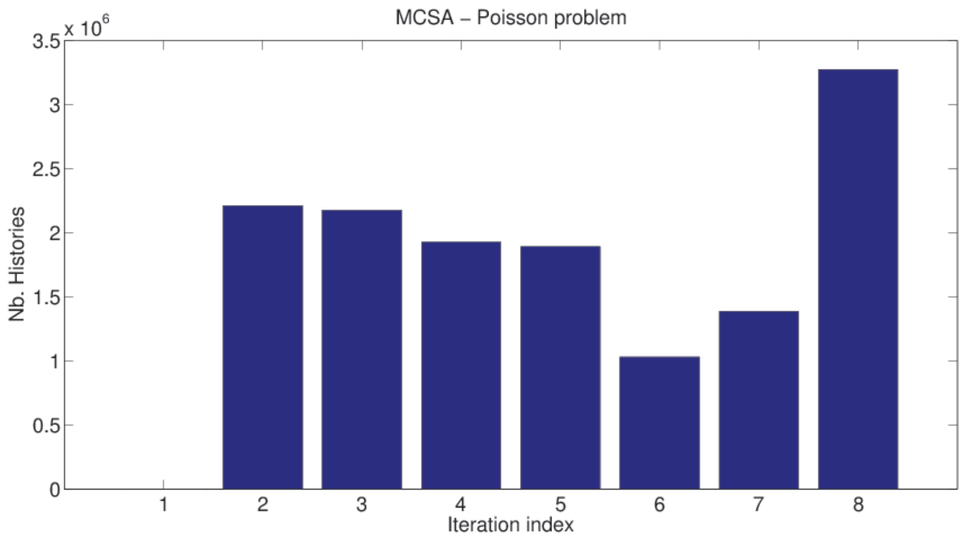

Figure 5.2: MCSA - Poisson problem. Number of random walks employed at each iteration. The number for the first iteration is 2000.

algorithm	relative err.	# iterations	average # histories per iteration
Richardson	$9.8081 \cdot 10^{-8}$	3133	-
Sequential MC	$7.9037 \cdot 10^{-8}$	9	8.264,900
Adjoint MCSA	$8.0872 \cdot 10^{-8}$	8	1,738,250

Table 5.14: Numerical results for the Poisson problem.

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

Table 5.15: Numerical results for the diffusion reaction problem.

Reaction-diffusion problem

Here we consider the simple reaction-diffusion problem

$$\begin{cases} -\Delta u + \sigma u = f \quad \text{in} \quad \Omega, \\ u = 0 \quad \text{on} \quad \partial\Omega, \end{cases} \tag{5.32}$$

where $\Omega = (0,1) \times (0,1)$, $\sigma = 0.1$ and $f \equiv 1$. A 5-point finite difference scheme is applied to discretize the problem. The number of nodes on each direction of the domain is 100, so that $h \approx 0.01$. The discretized problem is $n \times n$ with $n = 9604$. A left diagonal preconditioning is again applied to the coefficient matrix obtained from the discretization, which is strictly diagonally dominant. The 1-norm of the iteration matrix is $\|H\|_1 \approx 0.9756$. This automatically guarantees the convergence of the Adjoint Monte Carlo linear solver. In Table 5.15 a comparison between the deterministic Richardson iteration, Sequential Monte Carlo and MCSA is provided. The results are similar to those for the Poisson equation. The number of histories per iteration for Sequential Monte Carlo and MCSA is shown in Figures 5.3 and 5.4.

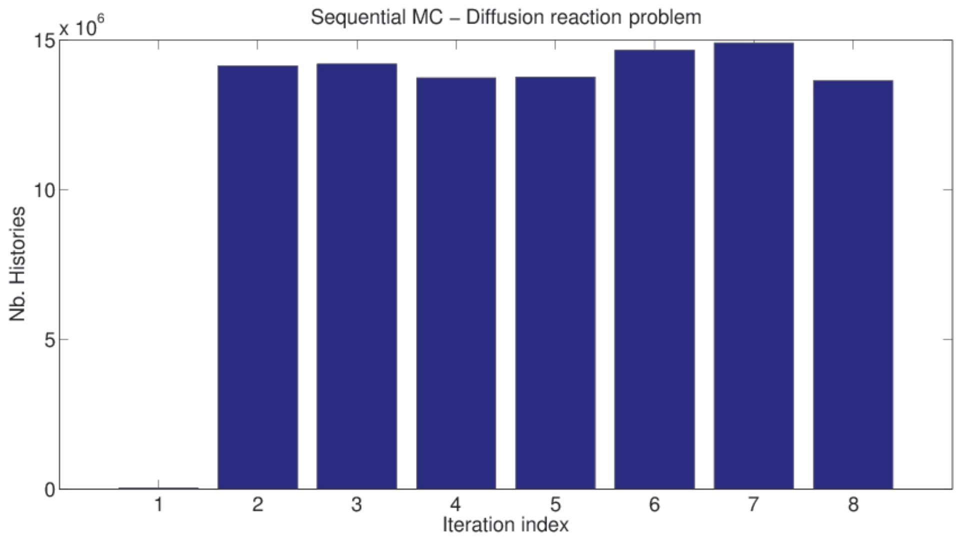

Figure 5.3: Sequential MC - Reaction-diffusion problem. Number of random walks employed at each iteration. The number first the first iteration is 36,000.

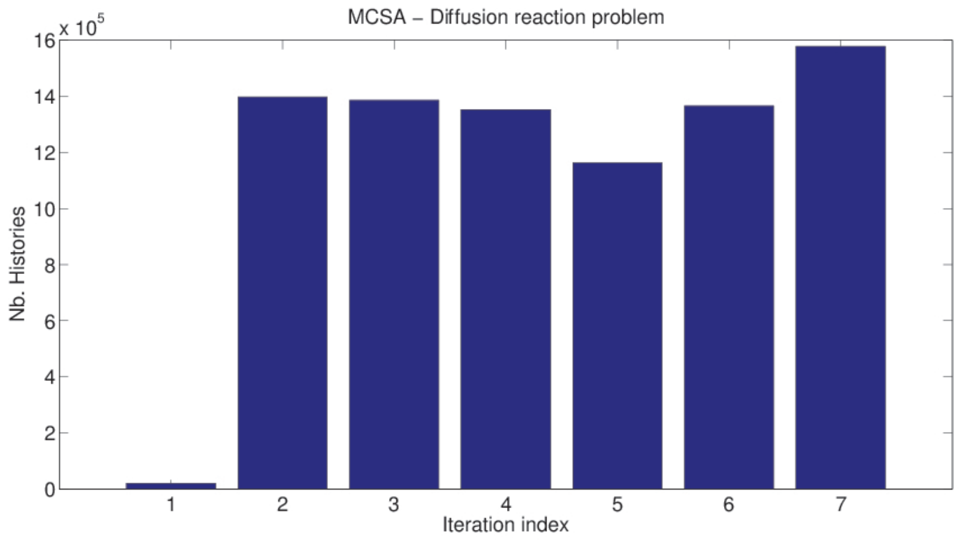

Figure 5.4: MCSA - Reaction-diffusion problem. Number of random walks employed at each iteration. The number first the first iteration is 36,000.

Parabolic problem

Here we consider the following time-dependent problem:

$$\begin{cases} \frac{\partial u}{\partial t} - \mu\Delta u + \boldsymbol{\beta}(\mathbf{x}) \cdot \nabla u = 0, & \mathbf{x} \in \Omega, \quad t \in (0, T] \\ u(\mathbf{x}, 0) = u_0, & \mathbf{x} \in \Omega \\ u(\mathbf{x}, t) = u_D(\mathbf{x}), & \mathbf{x} \in \partial\Omega, \quad t \in (0, T], \end{cases} \tag{5.33}$$

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

Implicit discretization in time (backward Euler scheme) with time step Δt and a spatial discretization with quadrilateral linear finite elements using the IFISS toolbox [22] leads to a sequence of linear systems of the form

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

Here we restrict our attention to a single generic time step. The right-hand side is chosen so that the exact solution to the linear system for the specific time step chosen is the vector of all ones. For the experiments we use a uniform discretization with mesh size $h = 2^{-8}$ and we let $\Delta t = 10h$. The resulting linear system has $n = 66,049$ unknowns.

We use the factorized sparse approximate inverse AINV [10] as a right preconditioner, with drop tolerance $\tau = 0.05$ for both inverse factors. With this choice, the spectral radius of the iteration matrix $H = I - AP^{-1}$ is $\rho(H) \approx 0.9218$ and the spectral radius of $\hat{H}$ for the Adjoint Monte Carlo is $\rho(\hat{H}) \approx 0.9148$. The MAO transition probability is employed. Resorting to a uniform probability in this case would have impeded the convergence, since in this case $\rho(\hat{H}) \approx 1.8401$. This is an

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

Table 5.16: Numerical results for the parabolic problem.

example demonstrating how the MAO probability can improve the behavior of the stochastic algorithm, outperforming the uniform one.

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

As before, the threshold for the check on the relative residual is set to $\varepsilon = 10^{-8}$, while the threshold for the adaptive selection of the random walks is set to $\varepsilon_1 = 0.1$. The results for all three methods are shown in Table 5.16. As one can see, both Sequential Monte Carlo and MCSA dramatically reduce the number of iterations with respect to the purely deterministic preconditioned Richardson iteration, with MCSA oupterforming Sequential Monte Carlo. Of course each iteration is now more expensive due to the Monte Carlo calculations required at each Richardson iteration, but we stress that Monte Carlo is an embarrassingly parallel method. Monte Carlo calculations are also expected to be more robust in the presence of faults, which is one of the main motivations for the present work.

Finally, Figures 5.5 and 5.6 show the number of Monte Carlo histories per iteration for Sequential Monte Carlo and for MCSA, respectively.

5.5 Conclusions and future work

In this chapter we have reviewed known convergence conditions for Monte Carlo linear solvers and established a few new sufficient conditions. In particular, we have determined classes of matrices for which the method is guaranteed to converge. The main focus has been on the recently proposed MCSA algorithm, which clearly outperforms

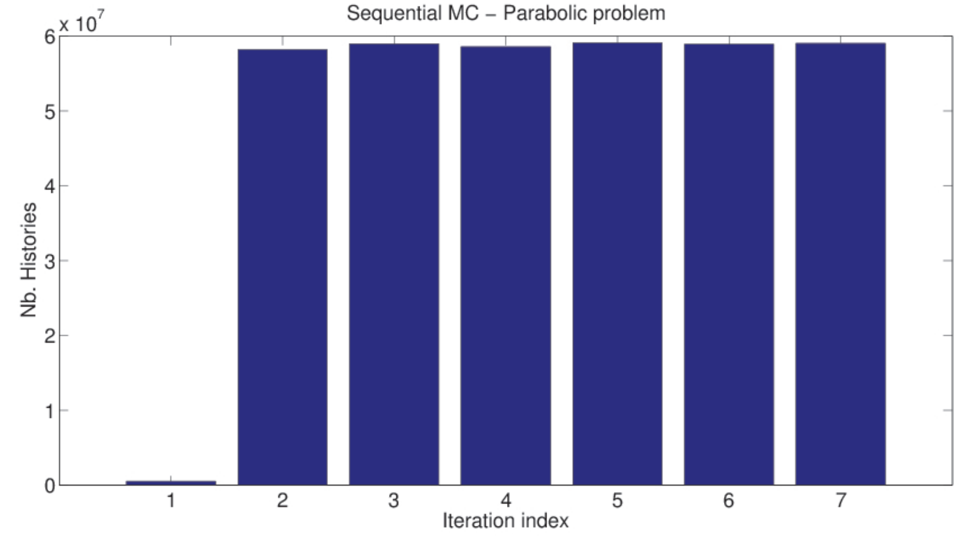

Figure 5.5: Sequential Monte Carlo - Parabolic problem. Number of random walks employed at each iteration. The number for the first iteration is 505,000.

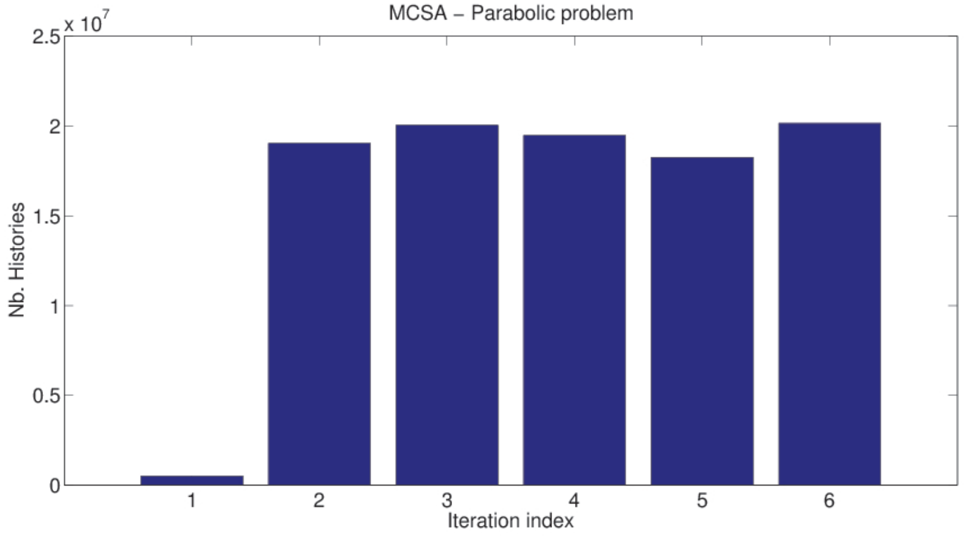

Figure 5.6: MCSA - Parabolic problem. Number of random walks employed at each iteration. The number for the first iteration is 505,000.

previous approaches. This method combines a deterministic fixed point iteration (preconditioned Richardson method) with a Monte Carlo acceleration scheme, typically an Adjoint Monte Carlo estimator. Various types of preconditioners have been tested, including diagonal, block diagonal, and sparse approximate inverse preconditioning.

Generally speaking, it is difficult to ensure a priori that the hybrid solver will converge. In particular, convergence of the underlying preconditioned Richardson iteration is necessary, but not sufficient. The application of hybrid solvers to non-diagonally dominant, steady-state problems presents a challenge and may require some trial-and-error in the choice of tuning parameters, such as the block size or drop tolerances; convergence can be guaranteed for some standard model problems but in general it is difficult to enforce. This is an inherent limitation of hybrid deterministic-stochastic approaches of the kind considered in this paper. It is of course possible in many cases to obtain convergence by combining the MCSA algorithm with a flexible Krylov subspace method like Flexible GMRES. However, this entails additional costs, decreased parallelism and increased storage requirements, as well as a possible reduction in the resilience of the algorithm due to the need for orthogonalization and the attendant additional communication needed.

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

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

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

Chapter 6

Remarks and future developments

In this thesis we explore different techniques to accelerate Richardson's iterations to solve sparse linear systems. The nature of the acceleration (deterministic or stochastic) varies according to the properties to be strengthened. Indeed, deterministic accelerations are adopted to obtain efficiency and scalability, whereas stochastic accelerations are meant to increase the resilience of the scheme.

Chapter 2 mainly describes standard techniques to solve sparse linear systems. A general discussion about two-level methods is provided, with particular attention to minimization techniques. This category includes the Anderson-Richardson method (AR), which alternates relaxation sweeps and Anderson mixing steps to update the approximate solution. AR has some properties in common with the Generalized Minimum Residual methods (GMRES). However, we disproved earlier statements in the literature showing that restarted ad truncated versions of AR differ from restarted and truncated versions of GMRES. Analogously to GMRES, AR struggles in addressing issues arising from new computational platforms as much as other Krylov methods. This is due to algebraic operations like inner products and leas-squares problem which involve global communications in a parallel computing framework.

Chapter 3 describes a method, namely Alternating Anderson-Richardson (AAR), that performs a deterministic acceleration of the Richardson's scheme based on the Anderson mixing. However, differently from Anderson-Richardson, AAR has the potential to significantly increase concurrency and reduce global communications via computational locality in a parallel environment. We conduct a theoretical analysis of AAR in exact arithmetic, comparing its convergence properties with the GMRES. These studies show that AAR is more robust against stagnation than other AR. This allows us to highlight the utility of computing multiple Richardson's steps without any extra computation in between. Firstly, they provide a fast way to identify a projection subspace where to solve the reduced problem and compute the acceleration. Secondly, they can efficiently damp high frequency components of the error in the same fashion as relaxation sweeps in multigrid algorithms. Lastly, performing multiple Richardson's steps without solving least-squares problems enhances the robustness against stagnation. Numerical results obtained in MATLAB for problem of different nature and with different preconditioners show that truncated variants of AAR are competitive against different versions of Restarted GMRES. Moreover, an augmented variant of the algorithm is proposed that guarantees convergence on positive definite matrices. This achievement is obtained without affecting the performance of AAR, as confirmed by numerical experiments at the end of the chapter. In the future, a deeper analysis of the geometric properties of the projection subspace identified by AAR is recommended. This could shed light to some inherent properties of the algorithm that could justify its use through a more rigorous geometric description.

Chapter 4 describes an implementation of AAR in `C` that employs up to 512 CPU cores to solve sparse linear systems with $\mathcal{O}(10^6)$ unknowns in an MPI environment. Numerical linear algebra operations are handled using the `PETSc` library. Two implementations of Truncated AAR(6,10) are tested. The first implementation solves the least-squares problem needed for the Anderson mixing via a parallel implementation of LSQR. The second version of AAR explicitly builds the normal equation associated with the least-squares problem and solves it independently on each process using the SVD decomposition. The performance of these two implementations of AAR is compared with Restarted GMRES(10) and Restarted GMRES(30) according to different preconditioning approaches. Although constructing the normal equation may severely impact the conditioning of the problem and thus the attainable accuracy, the numerical results show that this approach can be competitive against both the version of AAR using LSQR and Restarted GMRES. With regards to future developments, an objected oriented implementation of the algorithm is recommended, possibly combining distributed memory and GPU parallelization. In fact, the MPI environment could efficiently parallelize the least-square solving as the GPU parallelization may accelerate the computation of relaxation sweeps.

Chapter 5 deals with stochastic techniques to enhance resilience in stationary Richardson's iterations. The procedures presented in this chapter stochastically estimate the fixed point iterations via Monte Carlo samplings defined on random walks. In the context of Monte Carlo linear solvers (MCLS) we propose an adaptive procedure to select the number of statistical samplings needed to achieve a prescribed accuracy. These adaptive selection is based on the apparent relative standard deviation of the Monte Carlo estimator. The transition matrix adopted to define the random walks is strictly related to the iteration matrix of the deterministic fixed point scheme. However, statistical requirements over the finiteness of the expected value and the variance of the estimator impose stringent spectral conditions on the iteration matrix of the fixed point scheme to guarantee convergence. For instance, there are situations where the deterministic fixed point scheme is guaranteed to converge but its stochastic reinterpretation does not. This delicate situation makes the choice of the preconditioner crucial. On the one hand, the preconditioner must cap the fill-in effect since the iteration matrix must be explicitly built and stored. On the other hand, the preconditioner must be efficient enough to have the statistical requirements satisfied. To this goal, we generalize an earlier definition found in literature of block diagonally dominant matrices. According to the generalized definition that we adopt, a block diagonal preconditioner applied on a block diagonally dominant matrix would always guarantee the convergence of the Monte Carlo linear solver. Although not supported by any theoretical results, our numerical experiments show that sparse approximate inverse preconditioners easily accommodate the stochastic convergence criteria on parabolic partial differential equations. As for the future, the search of other classes of problems and preconditioners that could guarantee the convergence of MCLS is recommended.

Although the deterministic and stochastic accelerations analyzed in this thesis are employed to address different issues, a possible development may lead to their combination into a single iterative scheme. Indeed, convergence requirements for MCLS could be relaxed, since MC steps would be used as a preconditioner on an iterative scheme already accelerated via Anderson mixing. Currently, it is not clear whether one can avoid the explicit construction of the iteration matrix to employ MCLS. However, the use of block preconditioners may allow the distribution of this task across processors in an efficient and resilient manner. The final result can potentially lead to the construction of a linear solver that alternates Anderson and MC accelerations to simultaneously cope with inter-processor communications and system failures.

Bibliography

[1] E. Agullo, L. Giraud, A. Guermouche, J. Roman, and M. Zounon, *Towards resilient parallel linear Krylov solvers: recover-restart strategies*, Research Report N. 8324, June 2013, Project-Teams HiePACS, INRIA.

[2] V. Alexandrov, E. Atanassov, I. T. Dimov, S. Branford, A. Thandavan, and C. Weihrauch, *Parallel hybrid Monte Carlo algorithms for matrix computations problems*, Lecture Notes in Computer Science, 3516 (2005), pp. 752–759.

[3] D. G. Anderson, *Iterative Procedures for Nonlinear Integral Equations*, Journal of the Association for Computing Machinery, 12(4), 1965.

[4] H. Anzt, E. Chow, and J. Dongarra, *Iterative sparse triangular solves for preconditioning* Euro-Par 2015: Parallel Processing. Volume 9233 of the series Lecture Notes in Computer Science, pp. 650–661, 2015.

[5] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, UK, 1994.

[6] A. H. Baker, E. R. Jessup, and T. Manteuffel, *A Technique for accelerating the convergence of Restarted GMRES*, SIAM Journal on Matrix Analysis and Applications, 26(4), pp. 962–984, 2005.

[7] M. Benzi, *Preconditioning techniques for large linear systems: a survey*, Journal of Computational Physics, 182, pp. 417–477, 2002.

[8] M. Benzi, T. M. Evans, S. P. Hamilton, M. Lupo Pasini, S. R. Slattery, *Analysis of Monte Carlo accelerated iterative methods for sparse linear systems*, Numerical Linear Algebra with Applications, 24(3), e2088, 2017.

[9] M. Benzi, C. D. Meyer, and M. Tuma, *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM Journal on Scientific Computing, 17, pp. 1135–1149, 1996.

[10] M. Benzi and M. Tůma, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM Journal on Scientific Computing, 19, pp. 968–994, 1998.

[11] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, Academic Press, New York, 1979.

[12] A. Brandt, *Multi-Level Adaptive Solutions to Boundary-Value Problems*, Mathematics of Computation, 31, pp. 333–390, 1977.

[13] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial, Second Edition*, SIAM, 2000.

[14] A. Chapman, and Y. Saad, *Deflated and augmented Krylov subspace techniques*, Numerical Linear Algebra with Applications, 4(1), pp. 43–66, 1997.

[15] F. Chatelin, and W. L. Miranker, *Acceleration by aggregation of successive approximation methods*, Linear Algebra and Its Applications, 1982.

[16] R. Courant, K. Friedrichs, and H. Lewy, *Über die partiellen Differenzengleichungen der mathematischen Physik*, Mathematische Annalen, 100 (1928), pp. 32–74.

[17] J. H. Curtiss, *Sampling methods applied to differential and difference equations*, Proc. Seminar on Scientific Computation, Nov. 1949, IBM, New York, 1950, pp. 87–109.

[18] T. Davis, *The SuiteSparse Matrix Collection (formerly known as the University of Florida Sparse Matrix Collection)*, `http://www.cise.ufl.edu/research/sparse/matrices/`

[19] E. D. Dolan, and J. Moré, *Benchmarking optimization software with performance profiles*, Mathematical Programming, 91(2), pp. 201–213, 2002.

[20] I. T. Dimov and V. N. Alexandrov, *A new highly convergent Monte Carlo method for matrix computations*, Mathematics and Computers in Simulation, 47 (1998), pp. 165–181.

[21] I. T. Dimov, V. Alexandrov, and A. Karaivanova, *Parallel resolvent Monte Carlo algorithms for linear algebra problems*, Mathematics and Computers in Simulation, 55 (2001), pp. 25–35.

[22] H. C. Elman, A. Ramage, and D. J. Silvester, *IFISS: A computational laboratory for investigating incompressible flow problems*, SIAM Review, 56 (2014), pp. 261–273.

[23] M. Embree, *The tortoise and the hare restart GMRES*, SIAM Review, 45(2), pp. 259–266, 2003.

[24] T. M. Evans, S. W. Mosher, S. R. Slattery, and S. P. Hamilton, *A Monte Carlo synthetic-acceleration method for solving the thermal radiation diffusion equation*, Journal of Computational Physics, 258 (2014), pp. 338–358.

[25] H. Fang, and Y. Saad, *Two classes of multisecant methods for nonlinear acceleration*, Numerical Linear Algebra with Applications, 16, pp. 197–221, 2009.

[26] M. Fasi, J. Langou, Y. Robert, and B. Uçar, *A backward/forward recovery approach for the preconditioned conjugate gradient method*, Journal of Computational Science, 17, pp. 522–534, 2016.

[27] D. F. Feingold and R. S. Varga, *Block diagonally dominant matrices and generalizations of the Gerschgorin circle theorem*, Pacific Journal of Mathematics, 4 (1962), pp. 1241–1250.

[28] G. E. Forsythe and R. A. Leibler, *Matrix inversion by a Monte Carlo method*, Math. Tables Other Aids Comput., 6 (1952), pp. 78–81.

[29] S. Friedland and S. Karlin, *Some inequalities for the spectral radius of non-negative matrices and applications*, Duke Mathematical Journal, 42 (1975), pp. 459–490.

[30] M. Grote, and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM Journal on Scientific Computing, 18(3), pp. 838–853, 1997.

[31] G. H. Golub, and R. S. Varga, *Chebyshev Semi-Iterative Methods, Successive Over-Relaxation Iterative Methods, and Second-Order Richardson Iterative Methods, Parts I and II*, Numerische Mathematik, 3, pp. 147–156, pp. 157–168, 1961.

[32] L. Grigori, S. Moufawad, and F. Nataf, *Enlarged Krylov subspace conjugate gradient methods for reducing communication*, SIAM Journal on Matrix Analysis and Applications, 37(2), pp. 744–773, 2016.

[33] J. H. Halton, *Sequential Monte Carlo*, Mathematical Proceeding of the Cambridge Philosophical Society, 58 (1962), pp. 57–58.

[34] J. H. Halton, *Sequential Monte Carlo techniques for the solution of linear systems*, Journal of Scientific Computing, 9 (1994), pp. 213–257.

[35] J. Hao, M. Mascagni, and Y. Li, *Convergence analysis of Markov chain Monte Carlo linear solvers using Ulam-von Neumann algorithm*, SIAM Journal on Numerical Analysis, 51 (2013), pp. 2107–2122.

[36] M. R. Hestenes, and E. Stiefel, *Methods of Conjugate Gradients for Solving Linear Systems*, Journal of Research of the National Bureau of Standards, 49(6), 1952.

[37] M. Hoemmen, and M. Heroux, *Fault-tolerant iterative methods via selective reliability*, Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), Seattle, WA, vol. 3, IEEE Computer Society, 2011.

[38] M. Hoemmen, *Communication-avoiding Krylov Subspace Methods*, Doctoral Dissertation, University of California at Berkeley, Berkeley, CA, 2010.

[39] A. S. Householder, *The Theory of Matrices in Numerical Analysis*, Dover Publications, New York, 1964.

[40] L. Li, *On the iterative criterion for generalized diagonally dominant matrices*, SIAM Journal on Matrix Analysis and Applications, 24 (2002), pp. 17–24.

[41] J. Loffeld, and C. S. Woodward, *Considerations on the implementation and use of Anderson acceleration on distributed memory and GPU-based parallel computers*, Association for Women in Mathematics Research Symposium, Baltimore, MD, United States, 2015.

[42] J. Mandel, *On some two-level iterative methods*, Chapter: Defect Correction Methods, Computing Supplementum, Springer, 5, pp. 75–87, 1984.

[43] T. A. Manteuffel, *The Tchebychev Iteration for Nonsymmetric Linear Systems*, Numerische Mathematik, 28, pp. 307–327, 1977.

[44] R. B. Morgan, *A restarted GMRES method augmented with eigenvectors*, SIAM Journal on Matrix Analysis and Applications, 16, pp. 1154–1171, 1995.

[45] C. C. Paige, and M. A. Saunders, *Solution of Sparse Indefinite Systems of Linear Equations*, SIAM Journal on Numerical Analysis, 12, pp. 617–629, 1975.

[46] C. C. Paige, and M. A. Saunders, *LSQR: an algorithm for sparse linear equations and sparse least squares*, ACM Transactions on Mathematical Software (TOMS), 8(1), pp. 43–71, 1982.

[47] F. A. Potra, and H. Engler, *A characterization of the behavior of the Anderson acceleration on linear problems*, Linear Algebra and its Applications, 438(3), pp. 1002–1011, 2013.

[48] P. P. Pratapa, P. Suryanarayana, and J. E. Pask, *Anderson acceleration of the Jacobi iterative method: an efficient alternative to Krylov methods for large, sparse linear systems*, Journal of Computational Physics, 306, pp. 43–54, 2016.

[49] P. P. Pratapa, P. Suryanarayana and J. E. Pask, *Alternating Anderson-Richardson method: an efficient alternative to preconditioned Krylov methods for large, sparse linear systems*, arXiv preprint arXiv:1606.08740.

[50] K. Sargsyan, F. Rizzi, P. Mycek, C. Safta, K. Morris, H. Najm, O. Le Maitre, O. Knio, and B. Debusschere, *Fault resilient domain decomposition preconditioner for PDEs*, SIAM Journal on Scientific Computing, 37 (2015), pp. A2317–A2345.

[51] J. W. Ruge, and K. Stüben, *Algebraic multigrid*, Chapter: Multigrid methods, SIAM Frontiers in Applied Mathematics, SIAM, pp. 7–130, 1987.

[52] Y.Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, 2003.

[53] Y. Saad, and M. H. Schultz, *GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems*, SIAM Journal on Scientific and Statistical Computing, 7(3), pp. 856–869, 1986.

[54] Y. Saad, and K. Wu, *DQGMRES: a direct quasi-minimal residual algorithm based on incomplete orthogonalization* Numerical Linear Algebra with Applications, 3, pp. 329–343, 1996.

[55] S. R. Slattery, *Parallel Monte Carlo Synthetic Acceleration Methods For Discrete Transport Problems*, Ph.D. Thesis, University of Wisconsin-Madison, 2013, http://search.proquest.com/ebrary/docview/1449206072.

[56] S. R. Slattery, T. M. Evans, and P. P. H. Wilson, *A spectral analysis of the domain decomposed Monte Carlo method for linear systems*, International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering, Sun Valley Resort, ID, 2013.

[57] A. Srinivasan, *Monte Carlo linear solvers with non-diagonal splitting*, Mathematics and Computer in Simulation, 80 (2010), pp. 1133-1143.

[58] M. Stoyanov, and C. Webster, *Numerical analysis of fixed point algorithms in the presence of hardware faults*, SIAM Journal on Scientific Computing, 37(5), pp. C532–C553, 2015.

[59] E. de Sturler, H. A. van der Vorst, *Communication cost reduction for Krylov methods on parallel computers*, In: W. Gentzsch, U. Harms (eds) High-Performance Computing and Networking. HPCN-Europe 1994. Lecture Notes in Computer Science (LNCS), vol. 797. Springer, Berlin, Heidelberg.

[60] A. Toth, and C. T. Kelley, *Convergence analysis for Anderson acceleration*, SIAM Journal on Numerical Analysis, 53(2), pp. 805–819, 2015.

[61] H. F. Walker, and P. Ni, *Anderson Acceleration for fixed-point iterations* SIAM Journal on Numerical Analysis, 49(4), pp. 1715–1735, 2011.

[62] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.

[63] J. Xu, *Iterative Methods by Space Decomposition and Subspace Correction*, SIAM Review, 34, pp. 581–613, 1992.

[64] *LAPACK-Linear Algebra PACKage*. http://www.netlib.org/lapack/. Accessed: 2018-03-01.

[65] *Matrix Market Collection*, `http://math.nist.gov/MatrixMarket/`

[66] *PETSc/Tao*. https://www.mcs.anl.gov/petsc/. Accessed: 2018-03-01.

[67] *The Trilinos Project*. https://trilinos.org/. Accessed: 2018-03-01.