Solving linear systems arising from reservoirs modeling

Hussam Al Daas

SCIENCES SORBONNE UNIVERSITÉ

Inria inventeurs du monde numérique

Sorbonne Université

Inria

Doctoral School **École Doctorale Sciences Mathématiques de Paris Centre**

University Department **Inria**

Thesis defended by **Hussam Al Daas**

Defended on **13th December, 2018**

In order to become Doctor from Sorbonne Université

Academic Field **Applied Mathematics**

Speciality **Numerical Linear Algebra**

Solving linear systems arising from reservoirs modelling

Thesis supervised by	Laura Grigori	Supervisor
	Pascal Hénon	Co-Supervisor

Committee members

Referees	Serge Gratton	Professor at INP ENSEEIHT	
	Julien Langou	Professor at UC Denver	
Examiners	Frédéric Nataf	Senior Researcher at CNRS	Committee President
	Philippe Ricoux	Professor at MIT	
	Daniel Ruiz	Associate Professor at INP ENSEEIHT	
Supervisors	Laura Grigori	Senior Researcher at Inria	
	Pascal Hénon	Senior Researcher at TOTAL	

SCIENCES SORBONNE UNIVERSITÉ

Inria inventeurs du monde numérique

Sorbonne Université

Inria

Doctoral School **École Doctorale Sciences Mathématiques de Paris Centre**

University Department **Inria**

Thesis defended by **Hussam Al Daas**

Defended on **13th December, 2018**

In order to become Doctor from Sorbonne Université

Academic Field **Applied Mathematics**

Speciality **Numerical Linear Algebra**

Solving linear systems arising from reservoirs modelling

Thesis supervised by	Laura Grigori	Supervisor
	Pascal Hénon	Co-Supervisor

Committee members

Referees	Serge Gratton	Professor at INP ENSEEIHT	
	Julien Langou	Professor at UC Denver	
Examiners	Frédéric Nataf	Senior Researcher at CNRS	Committee President
	Philippe Ricoux	Professor at MIT	
	Daniel Ruiz	Associate Professor at INP ENSEEIHT	
Supervisors	Laura Grigori	Senior Researcher at Inria	
	Pascal Hénon	Senior Researcher at TOTAL	

Sorbonne Université

Inria

École doctorale **École Doctorale Sciences Mathématiques de Paris Centre**

Unité de recherche **Inria**

Thèse présentée par **Hussam Al Daas**

Soutenue le **13 décembre 2018**

En vue de l'obtention du grade de docteur de l'Sorbonne Université

Discipline **Mathématiques appliquées**

Spécialité **Algèbre linéaire numérique**

Résolution de systèmes linéaires issus de la modélisation des réservoirs

Thèse dirigée par Laura Grigori directrice
Pascal Hénon co-directeur

Composition du jury

Rapporteurs	Serge Gratton	professeur à l'INP ENSEEIHT	
	Julien Langou	professeur à l'UC Denver	
Examinateurs	Frédéric Nataf	directeur de recherche au CNRS	président du jury
	Philippe Ricoux	professeur au MIT	
	Daniel Ruiz	MCF à l'INP ENSEEIHT	
Directeurs de thèse	Laura Grigori	directrice de recherche à l'Inria	
	Pascal Hénon	directeur de recherche au TOTAL	

Keywords: krylov, block methods, inexact breakdown, deflation, recycling, domain decomposition

Mots clés : krylov, méthodes par bloc, inexact breakdown, déflation, recyclage, décomposition de domaine

This thesis has been prepared at the following research units.

Inria

2 rue Simone Iff
75012 Paris
France

☎ +33 1 44 27 42 98
Web Site http://www.inria.fr/

Laboratoire Jacques-Louis Lions

4 place Jussieu
75005 Paris
France

☎ +33 1 44 27 42 98
Web Site http://ljll.math.upmc.fr/

ACKNOWLEDGEMENTS

No matter how the eloquence harmonizes and combines words to communicate my gratitude to everyone who contributed to this work, it will be difficult to fulfil the goal. For all of you and from the bottom of my heart, I say thank you!

Foremost, I would like to express my sincere and limitless gratitude to my advisor, Laura Grigori, thank you very much for the opportunities you gave me. My work would not have succeeded without you. I deeply admire your perfectionism, patience, and knowledge. What I learned from you over the last four years at both professional and scientific levels push me to give my best. I am grateful to Pascal Hénon, who also supervised my thesis, he smoothed this work from its beginning and supported until its very end. The moments I spent in Pau were enriching and unforgettable.

Next, I would like to thank Julien Langou and Serge Gratton for reporting my manuscript.

It is a great honour that both of you as well as Frédéric Nataf, Daniel Ruiz, and Philippe Ricoux participate in the jury of this thesis.

I also pronounce my acknowledgements to Philippe Ricoux who believed in this work and subsidized it to be achieved. I will not forget Frédéric Nataf and the many discussions about domain decomposition that enriched my knowledge considerably. Many thanks to James Demmel for the research stay offered at UC Berkeley where I discovered the inner world of numerical computation.

My two dear friends, Olivier Tissot and Sebastien Cayrols, without you, these years would have been much harder. You are the best *co-bureau* ever. Our friendship is something I really appreciate. Oliver, where do I begin? With the scientific, the philosophical, or the linguistic discussions? Or should I start by thanking you for replacing me in the conferences that I could not attend? I could write pages thanking you for all the great (the not so great and, especially, the hardest) times where you were constantly supportive. Thank you, my friend. Sebastien, you do not know how the smile lost its way to our office after you left. I will always remember "the two minutes left" to finish the TSQR implementation. (For more than a week, they seemed endless.) Thank

you for the **Five minutes of C** lectures, they played an important role of finalizing the implementation part of this work. Furthermore, I cannot forget to thank Jan, Léa, Fabien R, Fabien W, Mathieu, Martin, Simplice, Alan, Bilal, Frédéric A, Noémie, Antoine, Chloé, Ani, Mehdi, and all my colleagues at LJLL and Inria-Paris especially who are at building A.

I also thank all the members of the team ALPINES whome I have not got to mention yet, Frédéric Hecht, Xavier Claeys, Pierre Marchand, Pierre-Henri Tournier, Marcela Bonazzoli, Axel Fourmont, and Ryadh Haferssas. I thank Pierre Jolivet as well for his advices and the discussions about Krylov methods.

To all those who prepared and facilitated my administrative papers, I thank you very much for your efforts, Laurence, Marion, and Ortenca from Inria; Catherine and Salima from LJLL; Monique from Total.

Of course there are my friends who were supportive on the social as well as the psychological levels, Soleiman, Racha, Bayram, Gontran, Stephan, and Chadi.

Nevertheless, if I go back in time to look for the reasons why I am here, I have no doubt that I will find it in my family home. There, where Science is sacred. That home contained unlimited love although it was small.

My father, I do not forget my first lesson of numerical computing. It took place in the car on the way Damascus to Palmyra about eleven years ago. You are my inspiring idol and the first reason for my passion in science. I wish you were alive, I really miss you.

My mother, whatever I do, I will never be able to give you back ε—of course for an arbitrary small $\varepsilon > 0$—of what you offered and still do. Although I wanted to be in your class at school, you did not want to. After many years, I realized that you were my first teacher in life. Thank you for your unconditional giving!

Hazem, Mawea, Ahmad, Sarah, Abboudeh, my lovely brothers and sisters, thank you for being always available and supportive despite the long distance between us. It has been more than eight years since the last time we were all together. I hope we can meet all soon.

Maria, Mazen, Julia, Reem, and Walid, the next generation, you are the sweetest angels I have ever seen.

I dedicate this thesis to my father, grandmother, grandfather, aunt and her son. RIP.

Finally, I would like to say thank you for Laura Mendoza who was next to me and supported me infinitely and unconditionally. You could stand my thesis symptoms of bad moods over the last two years. The long and cold *winter night* is etched in my memory. Without you the previous paragraphs would have been full of mistakes. Gracias por todo mi amor!

Solving linear systems arising from reservoirs modelling

Abstract

This thesis presents a work on iterative methods for solving linear systems that aim at reducing the communication in parallel computing. The main type of linear systems in which we are interested arises from a real-life reservoir simulation. Both schemes, implicit and explicit, of modelling the system are taken into account. Three approaches are studied separately. We consider non-symmetric (resp. symmetric) linear systems. This corresponds to the explicit (resp. implicit) formulation of the model problem. We start by presenting an approach that adds multiple search directions per iteration rather than one as in the classic iterative methods. Then, we discuss different strategies of recycling search subspaces. These strategies reduce the global iteration count of a considerable factor during a sequence of linear systems. We review different existing strategies and present a new one. We discuss the parallel implementation of these methods using a low-level language. Numerical experiments for both sequential and parallel implementations are presented. We also consider the algebraic domain decomposition approach. In an algebraic framework, we study the two-level additive Schwarz preconditioner. We provide the algebraic explicit form of a class of local coarse spaces that bounds the spectral condition number of the preconditioned matrix by a number pre-defined.

Keywords: krylov, block methods, inexact breakdown, deflation, recycling, domain decomposition

Résolution de systèmes linéaires issus de la modélisation des réservoirs

Résumé

Cette thèse présente un travail sur les méthodes itératives pour résoudre des systèmes linéaires en réduisant les communications pendant les calculs parallèles. Principalement, on est intéressé par les systèmes linéaires qui proviennent des simulations de réservoirs. Trois approches, que l'on peut considérer comme indépendantes, sont présentées. Nous considérons les systèmes linéaires non-symétriques (resp. symétriques), cela correspond au schéma explicite (resp. implicite) du problème modèle. On commence par présenter une approche qui ajoute plusieurs directions de recherche à chaque itération au lieu d'une seule direction comme dans le cas des méthodes classiques. Ensuite, on considère les stratégies de recyclage des espaces de recherche. Ces stratégies réduisent, par un facteur considérable, le nombre d'itérations global pour résoudre une séquence de systèmes linéaires. On fait un rappel des stratégies existantes et l'on en présente une nouvelle. On introduit et détaille l'implémentation parallèle de ces méthodes en utilisant un langage bas niveau. On présente des résultats numériques séquentiels et parallèles. Finalement, on considère la méthode de décomposition de domaine algébrique. Dans un environnement algébrique, on étudie le préconditionneur de Schwarz additif à deux niveaux. On fournit la forme algébrique explicite d'une classe d'espaces grossiers locaux qui bornent le conditionnement par un nombre donné *a priori*.

Mots clés : krylov, méthodes par bloc, inexact breakdown, déflation, recyclage, décomposition de domaine

Inria
2 rue Simone Iff – 75012 Paris – France

CONTENTS

INTRODUCTION

In this chapter, we give a brief introduction to the field of high performance scientific computing. We start by a short presentation of the history of computing. Then, we present the motivation for reducing communication methods. Furthermore, we present related methods that exist in the literature. Finally, we present the structure of this manuscript.

There were three necessities behind designing computers: performing arithmetical operations, reducing the time of computation, and, of course, eliminating the human liability to error. This design evolved over the history. The oldest known calculator, Abacus, goes back to as early as 2300 BC. The exact date of the invention of this tool is still unknown. During the 17^{th} century many mechanical devices were invented to do arithmetical operations. One of the most famous examples is the Napier's bones [37]. Since that time, a variety of devices have been constructed. They were not only able to do multiplication and division, but even extraction of square roots. The evolution of mechanical calculators passed by: the adding machine of Perrault (1666), of Caze (1720), and of Kummer (1847); Samuel Morland's multiplicator (1666); Michel Rous's instrument (1869); Léon Bollée's multiplying and dividing machine (1893); the arithmographs of Troncet and of Clabor (beginning of 20^{th} century); the arithmetical rods of Lucas and Genailles (beginning of 20^{th} century) [37]. In the thirties of the 20^{th} century, electromechanical computers were designed. It was Alan Turing who proposed the principles of modern computers [46]. Electronic computers have also their own evolution. It followed in somehow the electronic revolution when the transistor was invented. Arriving in 1976, the first supercomputer, called Cray-1, was installed at Los Alamos National Laboratory, USA. It used one vector processor. Six years later, the Cray X-MP was presented, the first supercomputer using shared memory parallel processors. In 1985, the first distributed memory multiprocessor supercomputer, Cray-2, saw the light. It was able to perform about 1.9×10^9 floating point operations per second (1.9 Giga flops) at its peak performance. Since that time, the race competition to construct the most powerful supercomputer began. Passing by Tera flops and Peta flops, supercomputers will achieve in the coming time the Exascale (predicted by 2021).

Over the last 42 years, the transistor counts followed an exponential growth predicted by Moore's Law. Till the beginning of the 21^{st} century, microprocessors had one

flop		Bandwidth	Latency
59 %	Network	26%	15%
	DRAM	23%	5%

Table 1: Annual improvement of the parameters of the *latency-bandwidth model* [16]. Network stands for the interconnection between processors on distributed memory architectures, DRAM is the reading access memory.

logical core. In order to have faster computing, companies had to increase the frequency. This one also increased exponentially. Consequently, the power consumption followed the exponential growth too. However, this would have become an unaffordable cost if the same strategy of increasing the frequency had been adapted. Thus, at that time, multi-core processors started to be produced. It was enough to wait two years to have a two times faster runtime of the same program on an up-to-date processor. Starting from 2005, this became different. To speedup the code, one should make it more parallel (suitable for multi-core architectures). On the other hand, distributed memory supercomputers have this necessity of parallelism. Since the beginning of distributed memory supercomputers, it was noticed that moving data between processors had a large impact on the performance of the code. Two parameters play the role in moving data. The latency of the network and the bandwidth of the network. Table 1 presents the annual improvement of the three previous parameters. Hence, to save time, we have to decrease the movement of data. This will be referred to as *reducing communication* or *minimizing communication* in the case of asymptotic achievement of the lower bound. Furthermore, the cost of energy ratio of moving data out of chip (DRAM, local interconnect or cross the system) to double precision floating point operation is about 100. Thus, to save energy, we have to reduce communication. To estimate the runtime of an algorithm, a performance model, called the *latency-bandwidth model*, was proposed in the literature. This model takes into account the time necessary to perform arithmetical operations, floating-point operations (flop) as well as to move data (latency and bandwidth).

In 1981 it was proved that the amount of communication necessary to perform sequential matrix-matrix multiplication has a lower bound [41]. In 2004 the proof was given for the distributed matrix-matrix multiplication [39]. Later in 2009, a generalization of these proofs was given for a wide variety of algorithms including LU factorization, Cholesky factorization, LDL^T factorization, QR factorization, algorithms for eigenvalues and singular values [8]. Different algorithms were redesigned to minimize communication such as LU and QR factorizations [17].

Scientific simulation, cloud computing, big data, etc., are applications that need algorithms that work efficiently on heterogeneous architectures[1]. In this thesis we are interested in solving linear systems arising from reservoir simulations. Such systems exhibit a sparse pattern of their non-zero elements, i.e., they have a small number of non-zero elements in their coefficient matrices. This is also the case in many other applications such as other physical simulations, big data and cloud computing. The well-

[1] Supercomputers using different types of processing units

known Gaussian elimination method (LU factorization) goes back to the 18^{th} century. However, this method, if applied to a sparse matrix, induces a fill-in in the factors L and U. Furthermore, the working flow of this method is hard to be parallelized. Since they depend only on matrix-vector multiplication and dot product operations, projection iterative linear solvers appeared to be more attractive for parallelization. They are also attractive for applications in which the coefficient matrix is not assembled (e.g., due to memory limitations) and only the application of the matrix operator on a vector is possible. Krylov iterative methods are widely used iterative linear solvers. The Conjugate Gradient method [35] is a Krylov method. It is one of the top ten algorithms of the last century [75].

As we mentioned previously, we are interested in solving linear systems. More precisely, our focus is on iterative linear solvers for solving linear systems arising from reservoirs simulations. Two formulation schemes exist to model the physical problem behind the reservoirs simulations, implicit and explicit schemes [11]. The induced linear systems are non-symmetric and symmetric, respectively. Generalized Minimal Residual (GMRES) [61], Generalized Conjugate Residuals (GCR) [60], Bi-Conjugate Gradient (BiCG) [60], Bi-Conjugate Gradient stabilized (BICGSTAB) [79], and Conjugate Gradient (CG) [35] are iterative methods for solving such linear systems. These methods are based on Krylov subspace iterations. The definition of the Krylov subspace related to the matrix $A \in \mathbb{C}^{n\times n}$ and the vector $b \in \mathbb{C}^n$ is:

$$K_k(A,b) = \text{span}\left\{b, Ab, \ldots, A^{k-1}b\right\}.$$

One iteration of a Krylov method consists of three main operations, a vector update of the form $y = \alpha x + \beta y$, a sparse matrix-vector multiplication SpMV, and one or multiple dot product operations. In terms of communication, the vector updates are performed locally. The sparse matrix-vector product induces point-to-point communication. However, the dot product is a global communication operation.

In order to reduce the communication, the global iteration count to reach convergence has to be reduced and/or the synchronization steps have to be hidden. To do so, different strategies were employed such as multiple search direction methods [27, 9, 29, 69], hiding communication methods [24, 23], communication avoiding methods [15], deflation methods [48, 20, 57], and preconditioning methods [60, 18, 70]. Multiple search direction strategies have interesting properties. They allow to approximate the solution of multiple right-hand sides simultaneously. The operations of such methods are of a type matrix-matrix rather than a vector-vector or a matrix-vector types in single search direction methods. Thus, they can benefit from efficient Basic Linear Algebra Subprograms 3 (BLAS 3). Most *simple* (i.e. one search direction) Krylov iterative methods have multiple search direction variants, usually referred to as block versions. Block GMRES [78], block CG [56], block BiCGSTAB [32] and other different block methods exist. Multiple direction methods can also be found in literature. Multi Preconditioned GMRES (MPGMRES) [27], Multiple Search Directions Conjugate Gradient (MSD-CG) [31], Enlarged CG (ECG), [29], and Multi Preconditioned CG (MPCG) [9, 69] are variants of *classic* Krylov methods that add multiple search directions at each iteration in order

to enhance the convergence of Krylov methods.

Avoiding communication and hiding communication methods rely on reformulating the algorithms of the *simple* Krylov methods in order to reduce synchronization steps per iteration. The former, which is referred to as *s*-step methods, was firstly introduced by Chronopoulos and Gear in [15] and revisited and discussed later in [36]. The latter is referred to as pipelined methods. Pipelined CG [24] and Pipelined GMRES [23] are two examples of such methods.

Since we are interested in non-symmetric linear systems, we mostly focus on the GMRES variants. Norm-minimizing iterative Krylov methods for non-symmetric matrices are long recurrence methods [21]. In practice and due to memory limitations, these methods must be restarted. This leads generally to slow rate of convergence. Different methods are suggested to maintain the efficiency of the full methods (no restart variant). Deflation strategies were suggested for that aim [20, 48]. Furthermore, when solving a sequence of linear systems arising from different applications (Newton's method, Constrained Pressure Residual preconditioner [80], shift and invert eigensolvers, etc.), consecutive coefficient matrices might have a small difference or even are equal. Hence, it is important to take advantage of previously solved linear systems in order to reduce the iteration count of later linear systems. A state of the art method was proposed by Parks et al. in [57].

Preconditioning the linear systems can reduce the iteration count and make the convergence fast. Efficient preconditioners usually depend on the underlined problem which makes their implementation more difficult and non-generic. Algebraic preconditioners that use only the coefficient matrix exist in the literature, such as incomplete factorizations methods and algebraic domain decomposition methods [60, 13]. Incomplete LU (ILU), incomplete Cholesky (IC) are widely known preconditioners [60]. Domain decomposition methods are usually used as preconditioners for Krylov subspace methods. The additive Schwarz method is naturally communication-avoiding (we refer the reader to [77, 18, 13] for a detailed overview on domain decomposition). Hence, this method can be a good candidate for reducing the communication in the iterative solver. However, this method is not robust with respect to the number of domains[2]. By increasing the number of domains the condition number might deteriorate [18]. To ensure robustness, different methods propose to add a coarse space (a supplementary domain) in order to maintain the robustness of the method. Generalized Eigenproblem in the Overlaps, GenEO, [70], FETI-GenEO [71], the coarse space based on the local Direchlet-to-Neumann maps [51], and other coarse spaces exist in literature. These coarse spaces are available for linear systems arising from the discretization of PDEs. They are efficient and not costly to be constructed but need access to the PDEs discretization information. This makes them not fully algebraic preconditioners.

This document is organized as follows. The following Chapter 1 presents state of the art strategies, methods and preconditioners related to iterative linear solvers. In Chapter 2 we introduce the enlarged GMRES method. This method is a (block) GMRES variant based on the enlarged Krylov subspace introduced in [29]. Since this method

[2] A domain can be considered to be associated with a processor

has a block scheme, we discuss the inexact breakdown [59] and we derive a non-costly detection test. We also discuss the deflation variant based on the strategy proposed by Erhel et al [20]. In Chapter 3, we discuss the solution of a sequence of linear systems. We start by deriving the deflation of singular vectors of the iteration matrix. In [57] the authors propose two types of deflation subspaces based on the Ritz eigenvectors of the iteration matrix A or its inverse A^{-1}. We review these deflation subspaces and we propose a deflation subspace based on the singular vectors of the iteration matrix. Due to convergence issues that might occur when the coefficient matrix changes, we propose a criterion based on the same deflation strategy to choose judiciously the deflation vectors to keep for deflating the new linear system. Afterwards, Chapter 4 presents our parallel implementation of the two methods presented in Chapters 2 and 3. We detail the basic linear algebra routines that we use in our implementation, the memory management, the structure of the solvers, the matrix free interface as well as the communication optimization. Numerical experiments up to 1024×4 cores are presented. Chapter 5 presents a class of efficient coarse spaces associated to the additive Schwarz preconditioner. We give an explicit algebraic formula of these spaces. These coarse spaces are efficient in the sense that they bound the condition number of the preconditioned matrix by a user pre-defined number with a minimal dimension coarse space. The coarse space introduced in [70] makes part of this class. Numerical experiments that illustrate the impact of the techniques described are discussed in each chapter.

This work led to the following

Journal paper

- H. Al Daas, L. Grigori, P. Hénon, Ph. Ricoux, Enlarged GMRES for solving linear systems with one or multiple right-hand sides [5, 4], Journal link, accepted in IMA Journal of Numerical Analysis, Journal link
- H. Al Daas, L. Grigori, A class of efficient locally constructed preconditioners based on coarse spaces [3] PDF, accepted in SIAM Journal on Matrix Analysis and Applications

Preprint

- H. Al Daas, L. Grigori, P. Hénon, Ph. Ricoux, Recycling of Krylov subspaces and reduction of deflation subspaces for solving sequence of linear systems [6], submitted to the ACM Transactions on Mathematical Software journal.

We note that the paper entitled "Enlarged GMRES for solving linear systems with one or multiple right-hand sides" corresponds to the Chapter 2, the paper entitled "A class of efficient locally constructed preconditioners based on coarse spaces" corresponds to the Chapter 5, and the preprint entitled "Recycling of Krylov subspaces and reduction of deflation subspaces for solving sequence of linear systems" corresponds to Chapters 3 and a part of Chapter 4.

CHAPTER 1

STATE OF THE ART

Outline of the current chapter

In this chapter, we present several variants of strategies and iterative linear solvers related to the Krylov subspace.

1.1 Othogonalization strategies

As we will see in the next section, the Arnoldi procedure constructs an orthogonal basis of the Krylov subspace. The Arnoldi procedure can be divided into three steps, matrix vector multiplication, orthogonalization of a vector against previous vectors, normalization of a vector. In this section, we discuss different methods to orthogonalize a set of vectors against another set of vectors as well as against each other. The Gram-Schmidt method is a well-known method to obtain an orthogonal basis vectors starting from a set of vectors. Algorithm 1 presents the classical Gram-Schmidt procedure. The classical Gram-Schmidt algorithm uses BLAS2 operations. However, due to finite precision arithmetic, round-off errors may affect the stability of the method. Two variants

Algorithm 1 Classical Gram-Schmidt procedure

Require: Set of linearly independent vectors $b_1, \ldots, b_m$
Ensure: $v_1, \ldots, v_m$ an orthogonal basis vectors of span$\{b_1, \ldots, b_m\}$
1: $v_1 = b_1 / \|b_1\|_2$
2: **for** $j = 1 : m-1$ **do**
3: $\quad w = b_{j+1}$
4: $\quad$ **for** $i = 1 : j$ **do**
5: $\quad\quad h_i = v_i^H w$
6: $\quad$ **end for**
7: $\quad w = w - \sum_{i=1}^{j} v_i h_i$
8: $\quad v_{j+1} = w / \|w\|_2$
9: **end for**

to the classical Gram-Schmidt exist and they are considered more numerically stable [25], namely, the classical Gram-Schmidt with reorthogonalization[1], Algorithm 3, and the modified Gram-Schmidt, Algorithm 2, procedures. The modified Gram-Schmidt

Algorithm 2 Modified Gram-Schmidt procedure

Require: Set of linearly independent vectors $b_1, \ldots, b_m$
Ensure: $v_1, \ldots, v_m$ an orthogonal basis vectors of span$\{b_1, \ldots, b_m\}$
1: $v_1 = b_1 / \|b_1\|_2$
2: **for** $j = 1 : m-1$ **do**
3: $\quad w = b_{j+1}$
4: $\quad$ **for** $i = 1 : j$ **do**
5: $\quad\quad h = v_i^H w$
6: $\quad\quad w = w - v_i h$
7: $\quad$ **end for**
8: $\quad v_{j+1} = w / \|w\|_2$
9: **end for**

procedure is less sensible to round off errors. However, it uses only BLAS1 operations. Moreover, in terms of communication, it is much more costly than the classical Gram-Schmidt. Indeed, at the j^{th} iteration of the loop, $j+1$ global communication are necessary to perform the dot product operations compared to 2 global communication in the case of the classical Gram-Schmidt. The double classical Gram-Schmidt can enhance the stability of the classical Gram-Schmidt while keeping the number of global communication constant at each iteration in the main loop of the procedure. Previous methods use BLAS1 and BLAS2 operations. Block variants that use BLAS3 operations exist. In the following we list the block variants of Gram-Schmidt as well as methods related to the orthogonalization of a set of small number of vectors referred to as *tall and skinny matrices*. The Cholesky QR algorithm [72] is an interesting candidate for the tall and

[1] We refer to this procedure as the double classical Gram-Schmidt

Algorithm 3 Double classical Gram-Schmidt procedure

Require: Set of linearly independent vectors $b_1, \ldots, b_m$
Ensure: $v_1, \ldots, v_m$ an orthogonal basis vectors of span$\{b_1, \ldots, b_m\}$
1: $v_1 = b_1/\|b_1\|_2$
2: **for** $j = 1 : m-1$ **do**
3: $w = b_{j+1}$
4: **for** $i = 1 : j$ **do**
5: $h_i = v_i^H w$
6: **end for**
7: $w = w - \sum_{i=1}^{j} v_i h_i$
8: **for** $i = 1 : j$ **do**
9: $h_i = v_i^H w$
10: **end for**
11: $w = w - \sum_{i=1}^{j} v_i h_i$
12: $v_{j+1} = w/\|w\|_2$
13: **end for**

skinny matrices. This method uses BLAS3 operations, and in terms of communication it

Algorithm 4 Cholesky QR algorithm

Require: Set of linearly independent vectors $b_1, \ldots, b_m$
Ensure: $v_1, \ldots, v_m$ an orthogonal basis vectors of span$\{b_1, \ldots, b_m\}$
1: let $P = (b_1, \ldots, b_m)$
2: compute $C = P^H P$
3: compute Cholesky factorization of C, $C = R^H R$
4: $V = PR^{-1}$
5: set $v_1, \ldots, v_m$ to be the columns of V

has only one synchronization step.

The TSQR algorithm [17] is a competitive candidate for the same application. It has the same communication cost of Cholesky QR. An important advantage of using TSQR is its numerical stability comparing to the Cholesky QR. However, it requires more computational costs. We refer the reader to [17] for more details on this method.

Householder and block Householder transformation are other methods that are used to orthogonalize a set of vectors, we refer the reader to [26, 64] for more details.

In the following we define the Krylov subspace related to a matrix $A \in \mathbb{C}^{n\times n}$ and a vector $b \in \mathbb{C}^n$

1.2 Krylov subspaces

Given a matrix $A \in \mathbb{C}^{n\times n}$ and a vector $b \in \mathbb{C}^n$, the Krylov subspace of dimension k related to A and b is defined as:

Algorithm 5 Block classical Gram-Schmidt procedure

Require: Set of linearly independent block of vectors $B_1,\dots,B_m = (B_1^1,\dots,B_1^{s_1}),\dots,(B_m^1,\dots,B_m^{s_m})$

Ensure: $V_1,\dots,V_m$ an orthogonal basis vectors of span$\left\{(B_1^1,\dots,B_1^{s_1}),\dots,(B_m^1,\dots,B_m^{s_m})\right\}$

1: QR factorization of B_1, $B_1 = V_1 R$
2: **for** $j = 1 : m-1$ **do**
3: $W = B_{j+1}$
4: **for** $i = 1 : j$ **do**
5: $H_i = V_i^H W$
6: **end for**
7: $W = W - \sum_{i=1}^{j} V_i H_i$
8: QR factorization of W, $W = V_{j+1} R$
9: **end for**

Algorithm 6 Modified block Gram-Schmidt procedure

Require: Set of linearly independent block of vectors $B_1,\dots,B_m = (B_1^1,\dots,B_1^{s_1}),\dots,(B_m^1,\dots,B_m^{s_m})$

Ensure: $V_1,\dots,V_m$ an orthogonal basis vectors of span$\left\{(B_1^1,\dots,B_1^{s_1}),\dots,(B_m^1,\dots,B_m^{s_m})\right\}$

1: QR factorization of B_1, $B_1 = V_1 R$
2: **for** $j = 1 : m-1$ **do**
3: $W = B_{j+1}$
4: **for** $i = 1 : j$ **do**
5: $H = V_i^H W$
6: $W = W - V_i H$
7: **end for**
8: QR factorization of W, $W = V_{j+1} R$
9: **end for**

$$K_k(A,b) = \text{span}\left\{b, Ab, \ldots, A^{k-1}b\right\}.$$

1.2.1 Preliminaries

Let $A \in \mathbb{C}^{n\times n}$ be a non-singular matrix and let P be a polynomial of degree $k \geq 0$ with coefficients $p_0, \ldots, p_k \in \mathbb{C}$, i.e.,

$$P(X) = p_0 + p_1 X + \ldots + p_k X^k.$$

The matrix $P(A) \in \mathbb{C}^{n\times n}$ is defined as:

$$P(A) = p_0 I_n + p_1 A + \ldots + p_k A^k,$$

where $I_n \in \mathbb{C}^{n\times n}$ is the identity matrix.

The minimal polynomial of A is defined as the polynomial P_A of minimal degree k_A such that $p_{k_A} = 1$ and $P_A(A) = 0_{n,n}$, where $0_{n,n} \in \mathbb{C}^{n\times n}$ is the zero matrix. Following the Cayley-Hamilton theorem, each matrix has a minimal polynomial of degree $k_A \leq n$. Note that the matrix A^{-1} can be expressed as a polynomial of A. Given a vector $b \in \mathbb{C}^n$, we can also define the minimal polynomial of the dual (A,b) as the polynomial $P_{A,b}$ of minimal degree $k_{A,b}$ such that $p_{k_{A,b}} = 1$ and $P_{A,b}(A)b = 0_n$. Note that $k_{A,b} \leq k_A$. We refer to $k_{A,b}$ as the grade of b with respect to A. An important remark is that the vector $A^{-1}b$ can be expressed explicitly as a polynomial of A applied on the vector b. Proposition 1 states general properties of the Krylov subspaces.

Proposition 1. *The following properties hold:*

- $dim(K_i(A,b)) = i$, *for all* $i \leq k_{A,b}$
- $K_{k_{A,b}}(A,b) = K_{i+k_{A,b}}(A,b)$, *for all* $i \geq 0$
- $AK_i(A,b)) \subset K_{i+1}(A,b)$, *for all* $i \geq 0$

Since the Krylov subspace of dimension k consists of all linear combinations of the vectors $b, Ab, \ldots, A^{k-1}b$, it contains all vectors of the form $P(A)b$ where P is a polynomial of degree $k-1$. This is one reason for which these subspaces play an important role in iterative methods to solve approximately linear system of the form

$$Ax = b. \tag{1.1}$$

Indeed, the approximate solution will be of the form $P(A)b$, where P is a polynomial of degree k such that $P(A)b \approx A^{-1}b$.

In the following section, we discuss the construction of basis vectors in the Krylov subspace.

1.2.2 Arnoldi's method

Arnoldi's method constructs a basis of the Krylov method. It was at first introduced by W. E. Arnoldi to solve eigenvalue problems [7]. The method can reduce a general square matrix to an upper Hessenberg form via unitary transformations. Algorithm 7 presents the basic algorithm of Arnoldi to construct a basis of the Krylov subspace. Two possible

Algorithm 7 Arnoldi procedure with classical Gram-Schmidt orthogonalization

Require: Matrix $A \in \mathbb{C}^{n\times n}$, vector $v_1 \in \mathbb{C}^n$ of norm 1, dimension of the Krylov subspace m

Ensure: V_m basis of the Krylov subspace $K_m(A, v_1)$, H_m Hessenberg matrix

1: **for** $j = 1 : m$ **do**
2: $w = Av_j$
3: **for** $i = 1 : j$ **do**
4: $h_{ij} = v_i^H w$
5: **end for**
6: $w = w - \sum_{i=1}^{j} v_i h_{ij}$
7: $h_{j+1,j} = \|w\|_2$, define the Hessenberg matrix $H_j \in \mathbb{C}^{(j+1)\times j}$, $H_j = (h_{ik})_{1\leq i\leq (j+1), 1\leq k\leq j}$
8: **if** $h_{j+1,j} = 0$ **then**
9: stop
10: **end if**
11: $v_{j+1} = w/h_{j+1,j}$
12: **end for**

outputs can result from Algorithm 7. Either the method stops at iteration $j_0 < m$ or it performs the m iterations. The first case means that the grade of v_1 is equal to j_0 and $K_m(A, v_1) = K_{j_0}(A, b)$. Hence, $V_m = \{v_1, \ldots, v_{j_0}\}$. In the second case the set of basis vectors of $K_m(A, b)$ is $V_m = \{v_1, \ldots, v_m\}$. Note that for $j < m + 1$ ($j < j_0 + 1$ in the case where the method stops) we have,

$$v_{j+1} h_{j+1,j} = Av_j - \sum_{i=1}^{j} h_{i,j} v_i,$$

where $V_j = \{v_1, \ldots, v_j\}$. Or in an equivalent way,

$$AV_j = V_{j+1} H_j.$$

1.3 Krylov subspace methods

In this section we review classical Krylov methods and several variants related to them.

1.3.1 Classical Krylov subspace methods

In this section, we review two widely known Krylov subspace methods, the Conjugate Gradient (CG) [35] and the Generalized Minimal Residual (GMRES) [61].

CG: [35] the Conjugate Gradient method is an iterative method that finds at iteration k the approximate x_k such that $\|x_* - x_k\|_A$ is minimal, where $\|\,.\,\|_A$ stands for the A-norm (A must be SPD) and x_* is the exact solution of the linear system. This method is one of the top ten algorithms of the last century. It has a short recurrence formulation that avoids storing the Krylov subspace vectors. Algorithm 8 presents the simple CG method.

Algorithm 8 CG

Require: Matrix $A \in \mathbb{C}^{n\times n}$(SPD), right-hand-side $b \in \mathbb{C}^n$, initial guess solution x_0, and the threshold of convergence ε
Ensure: Approximate solution x such that $\|b - Ax\|_2 < \varepsilon$

$r_0 = b - Ax_0$, $\rho_0 = \|r_0\|_2^2, j = 1$
while $\sqrt{\rho_{j-1}} > \varepsilon\|b\|_2$ **do**
  if $j == 1$ **then**
    $p = r_0$
  else
    $\beta = \rho_{j-1}/\rho_{j-2}$, $p = r + \beta p$
  end if
  $w = Ap$
  $\alpha = \rho_{j-1}/p^h w$
  $x = x + \alpha p$
  $r = r - \alpha w$
  $\rho_j = \|r\|_2^2$
  $j = j + 1$
end while

GMRES: [61] the Generalized Minimal Residual method is an iterative method that constructs an orthonormal basis of the Krylov subspace $K_k(A, b)$ at iteration k and finds the approximate solution x_k such that $r_k = b - Ax_k$ is of minimal norm (the Euclidean norm) over the Krylov subspace (A is general invertible matrix). This method has no short recurrence formulation. Thus, due to memory limitations, the method has to throw away the basis vectors of the Krylov subspace and restarts. That would lead in general to slow down the convergence. Algorithm 9 presents the simple GMRES method.
FGMRES: [62] the Flexible GMRES is a variant of GMRES that allows to use different preconditioners, each applied at one iteration. The disadvantage of this method is that the memory to store vectors is doubled with respect to the GMRES method.

In the following section we present a brief overview of different variants of the classical Krylov subspace methods.

1.3.2 Variants of classical Krylov subspace methods

Deflation and recycling methods

GMRES-DR: [48] the GMRES with Deflated Restarting is a variant of restarted GMRES

Algorithm 9 GMRES

Require: Matrix $A \in \mathbb{C}^{n\times n}$, right-hand-sides $b \in \mathbb{C}^n$ and initial guess solution x_0
Ensure: Approximate solution x_m
1: $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$, $v_1 = r_0/\beta$
2: **for** $j = 1 : m$ **do**
3: $w = Av_j$.
4: **for** $i = 1 : j$ **do**
5: $h_{ij} = v_i^H w$.
6: **end for**
7: $w = w - \sum_{i=1}^{j} v_i h_{ij}$
8: $h_{j+1,j} = \|w\|_2$
9: **if** $h_{j+1,j} == 0$ **then**
10: set $m = j$ and go to 14
11: **end if**
12: $v_{j+1} = w/h_{j+1,j}$
13: **end for**
14: Solve the least squares problem $y_m = \arg\min_{y\in\mathbb{C}^m} \|H_m y - \beta e_1\|_2$,
where $H_m \in \mathbb{C}^{(m+1)\times m}$ is an upper Hessenberg matrix with non-zero coefficients (h_{ij}), $e_1 = (1, 0, \ldots, 0)^\top \in \mathbb{C}^{m+1}$
15: $x = x_0 + V_m y_m$, where $V_m = \{v_1, \ldots, v_m\}$

in which a fixed number of the Ritz vectors related to the matrix A^{-1} and the Krylov subspace are updated at the end of each cycle[2]. The advantage of this method with respect to the restarted GMRES is that the deflation subspace maintains the efficiency of the non-restarted (referred to as full) method. This deflation subspace has a fixed dimension. It is updated every cycle. A disadvantage of this method is that it does not allow to make benefit of the deflation subspace to solve a linear system with the same matrix and different right-hand sides.

RD-GMRES: [20] the Deflated GMRES is a variant of restarted GMRES in which a number of the Ritz vectors related to the matrix A and the Krylov subspace are computed at the end of the cycle in order to precondition the matrix A in the following cycle. The deflation subspace in this method helps to maintain the efficiency of the full GMRES when the GMRES method is restarted. Since the deflation subspace is used as a preconditioner, this method allows to solve linear systems with different right-hand sides, each at a time. The major issue of this method is that the deflation subspace dimension might increase every cycle. This would lead to memory issues. A strategy for keeping the dimension fixed can remedy this problem, however, it induces a costly computation.

GCRO-DR: [57] the Generalized Conjugate Residual with Deflated Restarting is a generalization of the GMRES-DR in which any subspace can be used in deflation. This

[2] A cycle is the set of consecutive iterations between two restarts

method is typically used for solving a sequence of linear systems. In some configurations and depending on the sequence of linear systems and the choice of the deflation subspace, the convergence might be worse than the one achieved by GMRES-DR.

Avoiding communication and hiding communication methods

s-step CG: [15, 36] (later called Communication-Avoiding CG method) given a vector b, the method suggests computing the sequence of vectors $Ab, A^2b, \ldots, A^sb$ by applying the sparse matrix A s times iteratively. Since A is sparse, the communication occurs only with neighbors to compute this sequence. To orthonormalize these vectors which form the columns of a tall and skinny matrix, they use a communication-avoiding QR method such as tall and skinny QR (TSQR) [17], or Cholesky QR CholQR [72]. Theoretically, this method is equivalent to the CG method and avoids the synchronization steps during s iterations. However, numerical stability issues appear in finite precision arithmetic leading to convergence issues. Furthermore, combining preconditioning techniques with this method is quite difficult.

p-CG: [24] the pipelined CG is a CG variant in which additional vectors are introduced to the classical CG method. This variant removes the costly global communication from the standard CG algorithm by only performing a single non-blocking global communication per iteration. This global communication phase can be overlapped by the matrix-vector product, which typically only requires local communication. This method might suffer from numerical stability issues too.

Multiple search direction methods

The main issue of the following methods is the necessary storage of the basis vectors of the search subspace. In methods using multiple search directions to solve linear systems with one right-hand side [27, 9, 29], it is in general hard to know a priori if that would reduce sufficiently the iteration count with respect to the extra computation performed.

BGMRES: [78] is a block variant of GMRES that solves a linear system with multiple right-hand sides simultaneously with the same number of synchronization steps per iteration with respect to solving one right-hand side. Furthermore, the iteration count to reach convergence is typically reduced by using a large search subspace. The amount of computation is much larger, reaching a factor s for the dot product operations per iteration, where s is the number of right-hand sides.

IBBGMRES: [59] the Inexact Breakdown Block GMRES is a block GMRES variant in which only the necessary vectors are added to the search subspace. This method can remedy the storage issues in block GMRES.

MPGMRES: [27] Multi Preconditioned GMRES is a GMRES-like method in which multiple (two or more) preconditioners are applied simultaneously, while maintaining minimal residual optimality properties. To accomplish this, a block version of Flexible GMRES is used, but instead of considering blocks starting with multiple right hand sides, the method starts with the initial residual and grows the space by applying each of the preconditioners to all current search directions and minimizing the residual norm

over the resulting larger subspace. The inconvenient part in this method is that the dimension of the subspace might increase exponentially leading to storage issues. A strategy to make the dimension of the subspace increase reasonably was suggested.

MPCG: [9] Multi Preconditioned CG is a generalization of the standard CG that uses multiple preconditioners, combining them automatically in an optimal way. The algorithm may be useful for domain decomposition techniques and other problems in which the need for more than one preconditioner arises naturally. The main issue of this method is that it loses the short recurrence formulation of the classical CG.

ECG: [29] Enlarged CG is a CG variant that consists of enlarging the Krylov subspace by a maximum given number of vectors per iteration.

1.4 Preconditioners

A preconditioner of the matrix A is an invertible matrix M such that $M^{-1} \approx A^{-1}$ in some sense. Given the linear system (1.1), the preconditioned system by M can be defined in different ways. The left preconditioned linear system related to (1.1) is defined as:

$$M^{-1}Ax = M^{-1}b. \tag{1.2}$$

The right preconditioned linear system related to (1.1) is defined as:

$$AM^{-1}x = b. \tag{1.3}$$

Let x_*, x_*^l, and x_*^r be the solution of (1.1), (1.2), and (1.3), respectively. The solution of (1.1) and (1.2) are the same $x_* = x_*^l$, however, $x_* = M^{-1}x_*^r$. If the matrix and the preconditioner are both Hermitian Positive Definite HPD (Symmetric Positive Definite SPD in the real case), it would be more appropriate to keep the Hermitian (Symmetric) structure of the matrix in the preconditioned linear system. Hence, let $M = R^H R$ the Cholesky decomposition of M, the symmetric preconditioned linear system is defined as:

$$R^{-H}AR^{-1}x = R^{-H}b. \tag{1.4}$$

Krylov iterative methods might need a large number of iterations in order to converge. Using a preconditioner enhances the rate of convergence and make the Krylov methods more attractive. In this context, we can imagine reasonable constraints on the preconditioner such as the small cost of constructing M explicitly or implicitly and the reasonable cost of computing $M^{-1}v$ for any vector v. The following preconditioners are widely used in the context of Krylov iterative methods.

Incomplete LU: [60] the general Incomplete LU factorization (ILU) process of a sparse matrix A computes a sparse lower triangular matrix L and a sparse upper triangular matrix U so that the residual matrix $R = LU - A$ satisfies certain constraints, such as having zero entries in some locations. Roughly, the more L and U have fill-in, the more efficient the preconditioner is (with respect to iteration count to reach convergence). However, the preprocessing cost to compute the factors is higher when

the fill-in increases.

Incomplete Cholesky IC: [60] the Incomplete Cholesky factorization is the variant of the ILU factorization for symmetric definite positive (SPD) matrices.

Block Jacobi: [60] the block Jacobi preconditioner matrix stands for a block diagonal matrix whose non-zero entry (i, j) correspond to the same (i, j) entry of the matrix A. This preconditioner is communication avoiding. Each block diagonal or multiple diagonal blocks can be associated with one processor making it able to apply it on an arbitrary vector without any necessary information available in another processor. The weak point of this preconditioner is that its efficiency is less with respect to the increase in the number of diagonal blocks.

Algebraic additive Schwarz: [13, 60] the additive Schwarz is originally an iterative method to solve PDEs [77, 18]. From an algebraic point of view it can be considered as a generalization of the block Jacobi methods, where the diagonal blocks might have an overlap with their neighbor diagonal blocks. The same issues and advantages of block Jacobi apply.

CHAPTER 2

ENLARGED GMRES

Outline of the current chapter

We propose a variant of the GMRES method for solving linear systems of equations with one or multiple right-hand sides. Our method is based on the idea of the enlarged

Krylov subspace to reduce communication. It can be interpreted as a block GMRES method. Hence, we are interested in detecting inexact breakdowns. We introduce a strategy to perform the test of detection. Furthermore, we propose a technique for deflating eigenvalues that has two benefits. The first advantage is to avoid the plateau of convergence after the end of a cycle in the restarted version. The second is to have a very fast convergence when solving the same system with different right-hand sides, each given at a different time (useful in the context of Constrained Pressure Residual preconditioner).

2.1 Introduction

In this chapter, $A \in \mathbb{C}^{n\times n}$ is a nonsingular non-Hermitian matrix. Let the system of linear equations

$$AX = B, \tag{2.1}$$

where $X \in \mathbb{C}^{n\times s}$, and $B \in \mathbb{C}^{n\times s}$ is full rank, with $s \geq 1$ the number of right-hand sides. Here, we suppose that $s \ll n$. Block Krylov subspace methods are iterative schemes used to solve this type of linear systems of equations. They find a sequence of approximate solutions $X_1, \ldots, X_j$ respectively in the affine spaces $X_0 + \mathcal{K}_j(A, R_0)$, where X_0 is the initial guess, R_0 is the corresponding initial residual and

$$\mathcal{K}_j(A, R_0) = BlockSpan\left\{R_0, AR_0, \ldots, A^{j-1}R_0\right\} \subset \mathbb{C}^{n\times s}$$

is the j^{th} block Krylov subspace related to A and R_0.

Generalized Minimal RESidual (GMRES) [61], Conjugate Gradient (CG) (Hermitian case) [35], Conjugate Gradient Squared (CGS) [68] and Bi-Conjugate Gradient STABilized (BiCGStab) [79] are widely used Krylov subspace methods. They were all initially introduced in the simple case $s = 1$. An iteration of a simple Krylov method (i.e., $s = 1$) consists of a matrix-vector multiplication (BLAS2), dot products and update of vectors (BLAS1). In terms of high performance computing, these operations, especially the dot products, are constrained by communication (between processors or between levels of the local memory hierarchy) since the computation part becomes negligible when the number of processors increases. Thus, the block-type of Krylov methods were introduced. These schemes have three main advantages. Firstly, matrix-set-of-vectors operations are used (BLAS3). Secondly, the solution of multiple right-hand-sides are computed simultaneously. Lastly, a faster convergence can be achieved by using a larger search subspace. Generally, simple Krylov subspace methods have a block variant, (e.g., block GMRES [78], block BiCGStab [32], block CG [56]). However, one issue related to block methods is that there are few papers addressing the convergence analysis, while for the methods previously mentioned, for the case ($s = 1$), the literature is rich with such studies [60]. O'Leary [56], studies the convergence analysis of block conjugate gradient and presents an estimation of the error in the approximate solutions. In [66],

Simoncini and Gallopoulos generalize the theory of convergence presented in [56] to the block GMRES method. This generalization is restricted to the special case when the real part of the spectrum is positive definite.

The methods referred to as s-step methods e.g., [15, 36] are based on the idea of performing s iterations of the simple method without communication (s here is different from the number of right-hand sides that we noted above). For this, s basis vectors are computed by performing s matrix-vector multiplications, then these basis vectors are orthogonalized by using block operations. The matrix-vector products are performed without communication. This is possible due to data redundancy. Recently, the enlarged Krylov subspace approach was introduced in [29] along with a communication reducing conjugate gradient based on it. In order to enlarge the search subspace, the authors split the initial residual into multiple vectors. They construct the block Krylov subspace that is associated to the matrix A and the block of vectors that are obtained from the splitting of the initial residual. The Krylov subspace is contained in the enlarged one. Thus, in the worst cases, it converges, at least, with the necessary iterations for CG to converge. In [29], authors present promising results when the enlarged Krylov conjugate gradient is applied on linear systems that converge slowly with the simple conjugate gradient method.

Iterative methods that rely on a block version of Krylov subspace produce inexact breakdowns, which are related to a rank deficiency in the block residual or in the block of search directions, before reaching convergence [43, 22]. Different strategies to deal with this issue are presented in the literature [54, 22, 43, 59, 33, 10, 2]. To detect inexact breakdowns in block-like GMRES, a rank test has to be done at each iteration. In [59, 10, 2], the authors propose an inexact breakdowns detection test based on SVD factorization of the block residual in the block Krylov subspace. This strategy implies the solution of the least squares problem at each iteration in order to obtain the block residual in the block Krylov subspace, then it performs its SVD factorization. The dimension of the block Krylov residual increases linearly with the number of iteration. Different strategies to update a rank-revealing factorization exist in the literature. In [74, 45], given a rank-revealing factorization of a matrix M, the authors present how to update this factorization when several lines or columns are added, by concatenation, to the matrix M. In block Krylov methods, the matrix $\bar{R}_j$ (the matrix that is factored to detect inexact breakdowns at iteration j) is different for each j. This matrix represents the block residual by the basis vectors of the block Krylov subspace, $R_j = V_{j+1}\bar{R}_j$. This representation changes from one iteration to another. More precisely, the matrix $\bar{R}_{j-1}$ is not necessarily a sub-matrix of $\bar{R}_j$. We reformulate the relation between R_j and the basis vectors of the block Krylov subspace by using an update strategy of the Hessenberg matrix, see Section 2. This reformulation allows to update the rank-revealing factorization and thus reduces the cost of the detection test.

Solving large-scale linear systems of equations by a long-recurrence Krylov method may require restarting the method. This slows down its convergence. To avoid this issue it is common to use the deflation of eigenvalues [47, 20, 49]. Before restarting (block) GMRES, either Ritz values or harmonic Ritz values and the associated vectors

are computed to construct a deflation subspace.

In this chapter we focus on the GMRES scheme as presented in [60]. We introduce Enlarged GMRES method, referred to as EGMRES. This method is based on enlarged Krylov subspaces [29]. It is adapted for solving linear systems of equations with one or multiple right-hand sides. An *enlarging factor* EF is given as a parameter. At each iteration the dimension of the enlarged Krylov subspace increases by a number s_j between 1 and $s \times EF$, where s is the number of right-hand sides. This number s_j decreases over iterations. The dimension of the enlarged Krylov subspace stops increasing when the exact solution is contained in the enlarged Krylov subspace. The enlarged Krylov subspace contains the classical (block) Krylov subspace. EGMRES algorithm performs two global communication steps per iteration. The first corresponds to the orthogonalization of the new basis vectors against the previous vectors. The second is associated to the orthonormalization of the added basis vectors to the enlarged Krylov subspace. A point-to-point communication is necessary to perform the sparse matrix-matrix multiplication (SpMM). Therefore, EGMRES and GMRES have the same number of messages per iteration. The size of the messages becomes larger in EGMRES. However, EGMRES converges faster especially on challenging linear systems, see Section 2.6. Thus the number of global communication phases is reduced. In terms of arithmetics, EGMRES performs more floating point operations (flops) compared to GMRES. Nevertheless, we benefit from the efficiency of BLAS3 to perform these extra operations. In addition, this extra computation is overlapped with communication when it is possible.

The chapter is organized as follows. In Section 2.2, we give a brief discussion of existing variants of GMRES and its block version. We review exact and inexact breakdowns as introduced in [59]. We review the deflated Arnoldi procedure and the inexact breakdowns detection test that is proposed in [59].

In Section 2.3, we introduce Enlarged GMRES. In Section 2.4 we present a strategy to reduce the size of the block basis vectors. We reformulate the detection test of inexact breakdowns presented first in [59, 10]. This reformulation leads to the factorization of an $s \times s$ matrix rather than a matrix of dimension approaching $js \times s$, where s is the number of columns of the initial block residual R_0 and j is the iteration number. In addition, we show that this $s \times s$ matrix can be computed iteratively. Furthermore, we study a new strategy based on rank-revealing QR to reduce the size of the block in BGMRES-like methods. We show that the reduced basis is sufficient to achieve the same rate of convergence as when no reduction is done. We compare our strategy on a set of matrices to the existing approach that is based on SVD [59], and we show that they have approximately the same behavior.

We show experimentally that the enlarged Krylov subspace method approximates better the eigenvalues of the input matrix than the classical GMRES method for the same basis size. This basis is built with a smaller number of iterations for the enlarged Krylov subspace method, hence, it costs less communication. We use this property to deflate eigenvalues between restart cycles. For this purpose, we introduce a criterion based on both the approximated eigenvalue and the norm of the residual of the associated eigenvector.

We refer to the resulting method as Restarted Deflated Enlarged GMRES or RD-EGMRES. By using RD-EGMRES, we obtain a gain of a factor up to 7 with respect to GMRES in terms of iteration count on our set of matrices. We show numerically how the enlarged Krylov subspace method better approximates the eigenvalues.

In Section 2.5 we adapt EGMRES to be a Constrained Pressure Residual (CPR) solver. The CPR-EGMRES is a special linear solver for saturation-pressure systems that arise from simulations of reservoirs. Since we are interested in solving linear systems arising from simulations of reservoirs, we adapt EGMRES to be used as a CPR solver, the CPR solver was introduced in [80]. Such linear systems are formed by two coupled systems. We propose to solve the global system (referred to as the second level) by using Enlarged GMRES. The first level corresponds to solving a sub-system associated with the pressure variable. This sub-system is solved at each iteration. To solve it, unlike the common choice of algebraic multigrid proposed in [63], we introduce two practical strategies based on using RD-EGMRES. Thus, by using RD-EGMRES, we benefit from the approximation of eigenvectors to solve the linear system with multiple right-hand sides that are given each one at a time. The first strategy uses a fixed number of iterations without the necessity to reach the convergence threshold. The second strategy uses the threshold of convergence as a stopping criterion. Since a Krylov iterative method is not a linear operator in general, the first strategy requires the usage of a flexible variant in the global level. Note that the second strategy can be considered as a linear operator by reason of convergence (we suppose that the convergence threshold is small enough), hence, we do not need to use the flexible variant on the second level. We compare these strategies in the numerical experiments in Section 2.6. CPR-EGMRES reduces the number of iterations up to a factor of 2 compared to the ideal CPR-GMRES that solves the first level with a direct LU solver.

Numerical experiments are presented in Section 2.6. First, we present results to show that the more we increase the enlarging factor, the faster the method converges. Furthermore, we show that reducing the basis by using the new strategy is as efficient as the approach based on SVD. We compare two thresholds for the criteria of eigenvalues deflation. This comparison is done with different maximal dimensions of the enlarged Krylov subspace. Then, we show results for linear systems of equations with multiple right-hand sides, each given at a time. This is related to the CPR preconditioner that is used later. Finally, results for linear systems of equations with multiple right-hand sides, given all at one time, are presented.

2.2 Background

In this section, we review the block GMRES method, exact breakdowns and the deflated Arnoldi procedure.

2.2.1 Notations

Matlab notations are used in a block sense: $M(i,j)$ is the element in the block line i and the block column j of the block matrix M. $(M(i,j))_{i,j}$ represents a block matrix M whose block elements are $M(i,j)$. $\|.\|_F$ represents the Frobenius norm. Let $t > 0$ be the enlarging factor of the (block) Krylov subspace, and $T = ts$, the number of columns of the enlarged residual, where s is the number of right-hand sides. If $t = 1$ then, the enlarged Krylov subspace is identical to the (block) Krylov subspace. Let $s_j \leq s$ be the number of added vectors to the basis of the block Krylov subspace $\mathcal{K}_{j-1}(A, R_0)$ at iteration j, and $c_j = s - s_j$. $S_j = \sum_{i=1}^{j} s_i$ is the dimension of the block Krylov subspace $\mathcal{K}_j(A, R_0)$. We denote the cardinal by #. The identity matrix of size l is denoted by I_l. The matrix of size $l \times m$ with zero elements is denoted by $0_{l,m}$. A tilde over a matrix V, i.e., $\tilde{V}$, means that an inexact breakdowns detection is done and this matrix is not updated yet. A bar over a matrix V, i.e., $\bar{V}$, is the representation of V in the projection subspace and this representation is by the constructed basis. V^H represents the conjugate of V. $V^\top$ represents the transpose of V. R_j and R_j^E are the (block) residual and the enlarged residual at the iteration j respectively. Similarly, we note X_j and X_j^E the solution and the enlarged solution. $\oplus$ refers to the direct sum between orthogonal subspaces. Finally, we define the following notations: $\tilde{V}_{j+1} \in \mathbb{C}^{n \times s_j}$ denotes the matrix whose columns are the generated basis vectors at iteration j. $V_{j+1} \in \mathbb{C}^{n \times s_{j+1}}$ is the matrix whose columns are effectively considered, as added vectors to the basis of $\mathcal{K}_j(A, R_0)$, to get $\mathcal{K}_{j+1}(A, R_0)$. $\mathcal{V}_j = [V_1, \cdots, V_j] \in \mathbb{C}^{n \times S_j}$ denotes the matrix whose columns are the basis vectors of the block Krylov subspace $\mathcal{K}_j(A, R_0)$. $D_j \in \mathbb{C}^{n \times c_{j+1}}$ is the matrix whose columns span the subspace left aside in iteration j.

2.2.2 Block Arnoldi procedure and block GMRES

The block Arnoldi procedure (see Algorithm 10) is the main part of the BGMRES method. It is basically the orthogonalization process applied on the new basis vectors to get an orthonormal basis for the block Krylov subspace.

Algorithm 10 Block-Arnoldi (A, V_1, m)

Require: Orthogonal matrix $V_1 \in \mathbb{C}^{n\times s}$, matrix $A \in \mathbb{C}^{n\times n}$, number of iterations m.
Ensure: Orthonormal block basis vector. $\mathcal{V}_{m+1}$, block Hessenberg matrix $H_m \in \mathbb{C}^{(m+1)s\times ms}$.

1: **for** $j = 1 : m$ **do**
2: $W = AV_j$.
3: **for** $i = 1 : j$ **do**
4: $H(i, j) = V_i^H W$.
5: **end for**
6: $W = W - \sum_{i=1}^{j} V_i H(i, j)$.
7: QR Factorization of W, $W = V_{j+1}H(j + 1, j)$.
8: $\mathcal{V}_m = [V_1, \cdots, V_m]$, $\mathcal{V}_{m+1} = [\mathcal{V}_m, V_{m+1}]$, $H_m = (H(i, j))_{i,j}$.
9: **end for**

The block generalized minimal residual method (see Algorithm 11), BGMRES [78], is a Krylov subspace method. It finds a sequence of approximate solutions X_j, $j > 0$, for the system of linear equations $AX = B$. The residual norm $\|R_j\|_F$ is minimal over the corresponding block Krylov subspace

$$\mathcal{K}_j(A, R_0) = BlockSpan\{R_0, AR_0, \ldots, A^{j-1}R_0\}. \tag{2.2}$$

This method relies on building an orthonormal basis for the block Krylov subspace by using the block Arnoldi procedure. Once we build the basis, we solve a linear least squares problem in that subspace to obtain the solution.

Algorithm 11 BGMRES

Require: Matrix $A \in \mathbb{C}^{n\times n}$, right-hand-sides $B \in \mathbb{C}^{n\times s}$, initial solution X_0 and the number of iterations m.
Ensure: Approximate solution X_m.

1: $R_0 = B - AX_0 \in \mathbb{C}^{n\times s}$.
2: QR Factorization of R_0, $R_0 = V_1\Pi_0$.
3: Get $\mathcal{V}_{m+1}$ and H_m using Block-Arnoldi (A, V_1, m) (Algorithm 10).
4: Solve the least squares problem $Y_m = \underset{Y\in\mathbb{C}^{ms\times s}}{\arg\min}\|H_m Y - E_1\Pi_0\|_2$,
where $E_1 = (I_s, 0_{m,s})^\top \in \mathbb{C}^{js\times s}$.
5: $X_m = X_0 + \mathcal{V}_m Y_m$.

An algebraic relation holds at each iteration of the algorithm, $AV_j = \sum_{i=1}^{j+1} V_i H_j(i, j)$. It leads to the relation

$$A\mathcal{V}_j = \mathcal{V}_j H_j(1 : j, 1 : j) + V_{j+1}H_j(j + 1, j)E_j^\top, \tag{2.3}$$

where $\mathcal{V}_j = [V_1, \cdots, V_j]$, and $E_j = (0_{s,(j-1)s}, I_s)^\top \in \mathbb{C}^{js\times s}$. A detailed overview of block Krylov methods is given in [33].

2.2.3 Block Arnoldi and exact breakdown

Definition 1. *A subspace $\mathcal{S} \subset \mathbb{C}^n$ is called A-**invariant** if it is invariant under the multiplication by A, i.e., $\forall u \in \mathcal{S}$, $Au \in \mathcal{S}$.*

The importance of having an A-invariant subspace, for instance of dimension p, is that this subspace contains p exact eigenpairs if the matrix A is diagonalizable. In some cases, the matrix W, see (Line 6, Algorithm 10), is rank deficient. This occurs when an A-invariant subspace is contained in the Krylov subspace.

Definition 2. *An exact breakdown [59] is a phenomenon that occurs at the j^{th} iteration in the block Arnoldi procedure when the matrix W, at (Line 6, Algorithm 10), is rank deficient. The order of the exact breakdown at iteration j is the integer c_{j+1} verifying $c_{j+1} = s - rank(W)$ where s is the rank of V_1.*

The following lemma is the GMRES case of [60, Proposition 6.1, p. 158]. It illustrates the importance of the breakdown in GMRES, we give its proof for completeness.

Lemma 1. *In GMRES, when a breakdown occurs during an iteration j, the Krylov subspace*

$$\mathcal{K}_j(A, R_0) = Span\{V_1, \cdots, V_j\}$$

is an A-invariant subspace.

Proof. A breakdown in GMRES occurs during the iteration j when $H_j(j+1, j) = 0$. Thus, immediately from relation (2.3), we get $A\mathcal{V}_j = \mathcal{V}_j H_j(1:j, 1:j)$. It yields that the subspace $Span\{V_1, \cdots, V_j\}$ is A-invariant. □

However, in general, for the block Arnoldi procedure, an exact breakdown does not mean that there is an A-invariant subspace. For example, starting the algorithm with the initial block (u, Au), for any $u \in \mathbb{C}^n$, yields an exact breakdown in the first iteration. Nevertheless, the obtained subspace is not necessarily A-invariant. We recall several equivalent conditions related to the exact breakdown in Theorem 1. For the details and the proof see [59]. Let c_{j+1} denote the rank deficiency of W (Line 6, Algorithm 10), i.e., $c_{j+1} = s - rank(W)$, where s is the rank of V_1.

Theorem 1. *In the block GMRES algorithm, let X be the exact solution and R_j be the residual at iteration j. The conditions below are equivalent:*

1. *An exact breakdown of order c_{j+1} at iteration j occurs.*
2. *$dim\{Range(V_1) \cap A\mathcal{K}_j(A, R_0)\} = c_{j+1}$.*
3. *$rank(R_j) = s - c_{j+1}$.*
4. *$dim\{Range(X) \cap \mathcal{K}_j(A, R_0)\} = c_{j+1}$.*

Proof. See the proof in [59]. □

As our method is based on the **enlarged Krylov subspace**, it naturally inherits a block version of GMRES. In the next section, we review the theory of block GMRES method with deflation at each iteration (referred to as IBBGMRES-R) proposed by Robbé and Sadkane [59]. This method was then reformulated in a different way by Calandra et al. [10] (referred to as BFGMRES-S).

Deflated Arnoldi relation

Here, we review the derivation of the modified algebraic relations of the Arnoldi procedure, presented in e.g. [59, 10]. We follow the presentation in [10]. We recall that $V_{j+1} \in \mathbb{C}^{n \times s_{j+1}}$ is the matrix formed by the columns considered to be useful and thus, added to the basis $\mathcal{V}_j$ of the block Krylov subspace. $D_j \in \mathbb{C}^{n \times c_{j+1}}$ is the matrix whose columns span the useless subspace. The range of D_j is referred to as the deflated subspace. The decomposition of the range of the matrix $[\tilde{V}_{j+1}, D_{j-1}]$ into two subspaces is

$$Range([\tilde{V}_{j+1}, D_{j-1}]) = Range(V_{j+1}) \oplus Range(D_j), \tag{2.4}$$

with $[V_{j+1}, D_j]^H [V_{j+1}, D_j] = I_s$. The s_{j+1}-dimension subspace, spanned by the columns of V_{j+1}, is added to the block Krylov subspace. The other c_{j+1}-dimension subspace, spanned by D_j, is left aside. At the end of iteration j, we want the following relation to hold

$$A\mathcal{V}_j = [\mathcal{V}_{j+1}, D_j] H_j, \tag{2.5}$$

where the columns of D_j represent a basis of the deflated subspace after j iterations. The columns of $\mathcal{V}_{j+1}$, stand for a basis for the block Krylov subspace $\mathcal{K}_{j+1}$. We assume that this relation holds at the end of iteration $j-1$. Thus,

$$A\mathcal{V}_{j-1} = [\mathcal{V}_j, D_{j-1}] H_{j-1}. \tag{2.6}$$

Let us study the iteration j. First, we multiply A by V_j. Then, we orthogonalize against $\mathcal{V}_j$ and against D_{j-1}. A QR factorization of the result leads us to $\tilde{V}_{j+1}$. In matrix form that could be written in the following equation,

$$A\mathcal{V}_j = [\mathcal{V}_j, D_{j-1}, \tilde{V}_{j+1}] \tilde{H}_j. \tag{2.7}$$

$\tilde{H}_j$ has the form

$$\tilde{H}_j = \begin{pmatrix} H_{j-1} & N_j \\ 0_{s_j, s_{j-1}} & M_j \end{pmatrix} \tag{2.8}$$

where $N_j = [\mathcal{V}_j, D_{j-1}]^H A V_j \in \mathbb{C}^{(S_{j-1}+s) \times s_j}$ and $(AV_j - [\mathcal{V}_j, D_{j-1}] N_j) = \tilde{V}_{j+1} M_j$ is the QR factorization. To transform the relation (2.7) to the form in (2.5), let $Q_{j+1} \in \mathbb{C}^{s \times s}$ be a unitary matrix such that

$$[D_{j-1}, \tilde{V}_{j+1}] Q_{j+1} = [V_{j+1}, D_j], \tag{2.9}$$

then, we have

$$A\mathcal{V}_j = [\mathcal{V}_{j+1}, \mathcal{D}_j] \mathcal{Q}_{j+1}^H \tilde{H}_j, \tag{2.10}$$

where $\mathcal{Q}_{(j+1),j} = \begin{pmatrix} I_{S_j} & 0 \\ 0 & Q_{j+1} \end{pmatrix}$. Finally, we can write

$$A\mathcal{V}_j = [\mathcal{V}_{j+1}, D_j]H_j. \tag{2.11}$$

In conclusion, the deflation of the converged subspace requires finding the matrix Q_{j+1}. We will address this later in Section 2.4.1. The strategy to reduce the basis is based on this algebra. In the remaining of this section, we show the inexact breakdowns detection as presented in [59, 10].

2.2.4 Inexact breakdowns and subspace decomposition

In [59], the authors introduce exact and inexact breakdowns in the BGMRES-like methods. They define the inexact breakdown as the following.

Definition 3. *An inexact breakdown is a phenomenon that occurs when the matrix*

$$\begin{pmatrix} R_0 & AR_0 & \dots & A^m R_0 \end{pmatrix}$$

becomes almost rank deficient.

Detecting inexact breakdowns and deflating useless vectors leads to less computation and more memory for useful vectors. In [59], they propose two strategies to detect inexact breakdowns. The first is related to the rank of the block residual while the second is related to the rank of the block basis vectors. In the same paper, the analysis shows that in practice it is more likely to detect the rank deficiency of the block residual rather than the block basis vectors. In this chapter we are interested in the detection test related to the block residual. Here, we present the inexact breakdowns detection test. We follow the presentation introduced in [10]. We start from relation (2.9). Given the matrix $[D_{j-1}, \tilde{V}_{j+1}]$ and the block Krylov residual $\bar{R}_j \in \mathbb{C}^{(S_j+s)\times s}$, find Q_{j+1} such that V_{j+1} spans the subspace that has not converged of the block residual. Let $\bar{R}_j = U\Sigma W^H$ be the SVD factorization of the block Krylov residual. In [59], the authors decompose this factorization as

$$\begin{aligned} \bar{R}_j &= \begin{pmatrix} U_1 & U_2 \\ U_{s+} & U_{s-} \end{pmatrix} \begin{pmatrix} \Sigma_1 & \\ & \Sigma_2 \end{pmatrix} [W_1, W_2]^H \\ &= \begin{pmatrix} U_1 \\ U_{s+} \end{pmatrix} \Sigma_1 W_1^H + \begin{pmatrix} U_2 \\ U_{s-} \end{pmatrix} \Sigma_2 W_2^H, \end{aligned} \tag{2.12}$$

with $\|\Sigma_2\|_2 < \varepsilon_0$. The projection of the block residual $R_j \in \mathbb{C}^{n\times s}$ on the subspace perpendicular to $\mathcal{K}_j$ is given by

$$\begin{aligned} (I - \mathcal{V}_j \mathcal{V}_j^H) R_j &= [0, D_{j-1}, \tilde{V}_{j+1}] \bar{R}_j \\ &= [D_{j-1}, \tilde{V}_{j+1}][U_{s+}\Sigma_1 W_1^H + U_{s-}\Sigma_2 W_2^H]. \end{aligned}$$

The choice of the considered basis vectors from the linear combinations of the matrix $[D_{j-1}, \tilde{V}_{j+1}]$, relies on the idea that they should be related to the left singular vectors with singular values of Σ_1 or we can write

$$Range(V_{j+1}) = Range((I - \mathcal{V}_j \mathcal{V}_j^H) R_j W_1) = Range([D_{j-1}, \tilde{V}_{j+1}] U_{s+} \Sigma_1).$$

To find the matrix Q_{j+1}, it is sufficient to take the unitary factor of the QR factorization of $U_{s+} \in \mathbb{C}^{s \times s_j}$ and complete its columns to an orthonormal basis of $\mathbb{C}^{s \times s}$.

$$\begin{aligned} Q_{j+1} &= qr(U_{s+} \Sigma_1) \\ &= qr(U_{s+}). \end{aligned}$$

The detection test of inexact breakdowns is done at every iteration. Hence, an SVD factorization of $\bar{R}_j \in \mathbb{C}^{(S_j+s) \times s}$ occurs at each iteration. During a cycle, the size of this problem grows linearly with the iteration number. In [74, 45] the authors update a rank-revealing factorization of a matrix M after adding several lines (or columns) to the matrix M by concatenation. However, the matrix $\bar{R}_j$ changes entirely from one iteration to another. In Section 2.4.1, we reformulate the relation between the j^{th} block residual R_j and the basis vectors of the block Krylov subspace by using the update of the QR factorization of the Hessenberg matrix. As a consequence, we obtain a rank-revealing update strategy that avoids solving the least squares problem in order to compute $\bar{R}_j$. Furthermore, a factorization of an $s \times s$ matrix is sufficient to detect the inexact breakdowns rather than performing an SVD factorization of a matrix of dimension approaching $js \times s$, see Proposition 3. In addition, a study of inexact breakdowns detection based on rank revealing QR is presented.

2.3 Enlarged GMRES

We introduce in this section EGMRES our new block GMRES method Enlarged GMRES which is based on enlarging the block Krylov subspace [29]. Indeed, for each one of the s right-hand sides, we add at each iteration multiple new basis vectors to the subspace. At the end, the obtained search subspace contains the original block Krylov subspace. We describe briefly how the enlarged Krylov subspace is obtained by using projection operators, thus reformulating the derivation from [29]. This method depends on the partition of the set of unknowns which is obtained by partitioning the graph of the matrix by using K-way partitioning [42].

Let $\zeta = \{1, \cdots, n\}$. We partition this set in t disjoint non trivial subsets denoted (ζ_i) with $i = 1, \cdots, t$. To each subset, we associate a projector P_i, such that

$$P_i : \mathbb{C}^{n \times s} \rightarrow \mathbb{C}^{n \times s} \tag{2.13}$$

$$u \rightarrow Z_i Z_i^H u, \tag{2.14}$$

where the j^{th} column of $Z_i \in \mathbb{R}^{n\times\#(\zeta_i)}$ is the $\zeta_i(j)^{th}$ canonical basis vector. Two properties follow

$$u = \sum_{i=1}^{t} P_i(u), \quad \forall u \in \mathbb{C}^{n\times s}; \tag{2.15}$$

$$P_i \perp P_j, \quad i \neq j. \tag{2.16}$$

Before defining the enlarged Krylov subspace, we define the enlarged residual using the projector P. We suppose that $\forall i \in \{1,\cdots,t\}$, $\forall j \in \{1,\cdots,s\}$, $\|P_i(R_0)(:,j)\|_2 \neq 0$. Thus, the enlarged residual is the matrix

$$P(R_0) = [P_1(R_0),\cdots,P_t(R_0)]. \tag{2.17}$$

Figure 2.1 illustrates the operator P, relation (2.17).

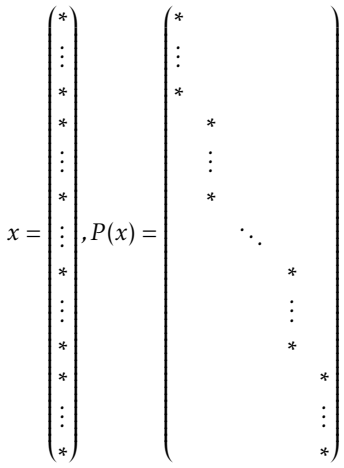

Figure 2.1: Illustration of enlarging a vector x.

Definition 4. *Let the system of linear equations $AX = B$, where $A \in \mathbb{C}^{n\times n}$ is nonsingular, $B \in \mathbb{C}^{n\times s}$ is full rank and $X \in \mathbb{C}^{n\times s}$. The j^{th} t-enlarged Krylov subspace associated with the matrix A and the initial residual R_0 is defined by*

$$\mathcal{K}_{j,t}(A,R_0) = BlockSpan\{P(R_0), AP(R_0),\cdots,A^{j-1}P(R_0)\}, \tag{2.18}$$

where the projection P is given by (2.17).

We will refer to the enlarged Krylov subspace $\mathcal{K}_{j,t}(A,R_0)$ by $\mathcal{K}_{j,t}$ when there is no ambiguity. For more details on the enlarged Krylov subspaces we refer the reader to [29]. While in [29] the authors focused on SPD matrices, in the following we derive an enlarged Krylov subspace method for solving linear systems of equations of general matrices.

Definition 5 (Enlarged GMRES)**.** *The Enlarged GMRES, denoted EGMRES, is an enlarged Krylov subspace method. It finds a sequence of approximate solutions* $\{X_1, \cdots, X_m\}$ *for the system of linear equations* $AX = B$*.* $X_j - X_0$ *belongs to the* j^{th} *enlarged Krylov subspace* $\mathcal{K}_{j,t}(A, R_0)$ *with* R_0 *the initial residual.* $\|R_j\|_F = \|B - AX_j\|_F$ *is minimal over the enlarged Krylov subspace.*

2.3.1 Enlarged GMRES algorithm

The following algorithm is the basic form of Enlarged GMRES. Let $\mathbf{1}_t$ be as

$$\mathbf{1}_t = [I_s, \cdots, I_s]^\top \in \mathbb{C}^{T \times s}, \tag{2.19}$$

where I_s the identity matrix of size s.

Algorithm 12 EGMRES

Require: Threshold of convergence ε_0, initial solution X_0.
Ensure: Approximate solution X_j.
1: $R_0 = B - AX_0$.
2: Form the enlarged residual $P(R_0)$ as in (2.17).
3: $R_0^E = P(R_0)$.
4: QR factor R_0^E, $R_0^E = V_1 \Pi_0$.
5: Set $E_0 = \Pi_0$ and $G_0 = 0_{T,T}$.
6: **for** $j = 1$ till convergence **do**
7: $W = AV_j$.
8: Orthogonalization procedure to get V_{j+1} and updating H_j and its QR factors $H_j = \mathcal{F}_j \begin{pmatrix} C_j \\ 0_{T,jT} \end{pmatrix}$ where $\mathcal{F}_j \in \mathbb{C}^{(j+1)T \times (j+1)T}$ and $C_j \in \mathbb{C}^{jT \times jT}$.
9: Compute $\begin{pmatrix} E_j \\ G_j \end{pmatrix} = \mathcal{F}_j^H \Pi_j$ where $\Pi_j = \begin{pmatrix} \Pi_0 \\ 0_{jT,T} \end{pmatrix}$, $E_j \in \mathbb{C}^{jT \times T}$ and $G_j \in \mathbb{C}^{T \times T}$.
10: **if** $\|G_j \mathbf{1}_t\|_F < \varepsilon_0$ **then**
11: Break.
12: **end if**
13: **end for**
14: Solve the linear least squares problem $Y_j = \underset{Y \in \mathbb{C}^{jT \times T}}{\arg\min} \|\Pi_j - H_j Y\|$, $Y_j = C_j \setminus E_j$.
15: $X_j = X_0 + [V_1, \cdots, V_j] Y_j \mathbf{1}_t$.

The update of the Hessenberg matrix (Line 8, Algorithm 12) means updating its QR factors $\mathcal{F}_j$ and C_j such that $H_j = \mathcal{F}_j \begin{pmatrix} C_j \\ 0_{T,jT} \end{pmatrix}$, where $\mathcal{F}_j \in \mathbb{C}^{(j+1)T \times (j+1)T}$ and $C_j \in \mathbb{C}^{jT \times jT}$.

In the following we prove that the EGMRES method finds the approximate solution X_j at iteration j such that the residual R_j has minimal Frobenius norm over the enlarged Krylov subspace $\mathcal{K}_{j,t}(A, R_0)$.

Proposition 2. *Following the notations in algorithm 12 we have*

$$\|B - AX_j\|_F = \min_{Y \in \mathbb{C}^{jT \times T}} \|\Pi_j \mathbf{1}_t - H_j Y \mathbf{1}_t\|_F.$$

Proof. We have

$$\begin{aligned} \|B - A(X_j + X_0)\|_F &= \|R_0^E \mathbf{1}_t - AX_j\|_F \\ &= \|V_1 \Pi_0 \mathbf{1}_t - [V_1, \cdots, V_{j+1}] H_j Y_j \mathbf{1}_t\|_F \\ &= \|\Pi_j \mathbf{1}_t - H_j Y_j \mathbf{1}_t\|_F \end{aligned}$$

By construction, Y_j minimizes the Frobenius norm of $\|\Pi_j \mathbf{1}_t - H_j Y \mathbf{1}_t\|_F$, where $Y \in \mathbb{C}^{jT \times T}$. Thus,

$$\|B - A(X_j + X_0)\|_F = \min_{Y \in \mathbb{C}^{jT \times T}} \|\Pi_j \mathbf{1}_t - H_j Y \mathbf{1}_t\|_F.$$

□

After the presentation of the EGMRES method, we remark that once we enlarge the block residual, it returns to block GMRES scheme. More precisely, let $b \in \mathbb{R}^{n \times s}$ be a set of vectors. In exact arithmetic, the operations at iteration i of EGMRES when solving $Ax = b$ are the same as the operations performed at iteration i of block GMRES when solving $Ax = P(b)$, where P is the enlarging operator that is defined in relation (2.17). Two differences exist between these two algorithms. The stopping criterion changes as shown in Proposition 2. EGMRES achieves convergence when the norm of the enlarged residual multiplied by the matrix $\mathbf{1}_t$, defined in relation (2.19), is less than the convergence threshold. In block GMRES the convergence is achieved when each vector of the block residual has a norm less than the convergence threshold. We note that the stopping criterion of block GMRES is stronger than the stopping criterion of EGMRES. The solution in EGMRES is recovered by multiplying the vectors of x by $\mathbf{1}_t$, while the solution vectors in block GMRES correspond to the vectors of right-hand sides respectively.

2.4 Inexact breakdowns and eigenvalues deflation in block GMRES

As mentioned previously, EGMRES and block GMRES do the same operations during each iteration if we consider the enlarged residual as an initial block residual. For this, in this section we study block GMRES rather than EGMRES. The application of this study on EGMRES is direct.

Relations in Lemma 2 and Proposition 3 hold until the end of this chapter.

2.4.1 Inexact breakdowns

In this section we reformulate the inexact breakdowns detection test in order to reduce its cost. In [59, 10], the authors propose a strategy to detect inexact breakdowns related to the block residual. We showed in Section 2.2 how this strategy adds only useful vectors to the block Krylov subspace. However, it needs to do an SVD factorization of the matrix representing the block Krylov residual $\bar{R}_j \in \mathbb{C}^{(S_j+s)\times s}$. Rank-revealing update strategies exist in the literature [74, 45]. When several lines (or columns) are concatenated to a matrix M an update of its rank-revealing factorization is not costly. However, the block residual in block GMRES changes entirely every iteration and has a dimension that depends on the iteration number j. Proposition 3 is the key idea to reduce the dimension of the SVD problem. Before that we need the following lemma 2. This lemma is going to be a tool in the remaining of this section.

Lemma 2. *The QR factorization of the matrix $\tilde{H}_j$ in the relation (2.7) is given by the relation*

$$\tilde{H}_j = \left(\prod_{i=0}^{j-2} \mathcal{Q}_{(j-i),j}^H\right)\left(\prod_{i=1}^{j} \mathcal{F}_{i,j}\right)\begin{pmatrix} C_j \\ 0_{s,S_j}\end{pmatrix}, \tag{2.20}$$

where

$$\mathcal{Q}_{i,j} = \begin{pmatrix} I_{S_{i-1}} & & \\ & Q_i & \\ & & I_{(S_j-S_{i-1})}\end{pmatrix}, \mathcal{F}_{i,j} = \begin{pmatrix} I_{S_{i-1}} & & \\ & F_i & \\ & & I_{(S_j-S_i)}\end{pmatrix}$$

and $C_j \in \mathbb{C}^{S_j\times S_j}$ is triangular. $Q_i \in \mathbb{C}^{s\times s}$ is the rotation matrix obtained by the inexact breakdowns test. $F_i \in \mathbb{C}^{(s_i+s)\times(s_i+s)}$ is the Householder transformation matrix used to triangularize the block $\tilde{H}_i(i:i+1,i)$ after updating $\tilde{H}_i(1:i,i)$ (2.7) by using F_k for $k = 1,\cdots,i-1$. By convention $S_0 = 0$.

Proof. Proof by induction. The case $j = 1$ corresponds to a basic Householder QR factorization.
Suppose that the relation holds for j. Let us prove it for $j+1$. We have in (2.8),

$$\tilde{H}_{j+1} = \begin{pmatrix} H_j & N_{j+1} \\ 0_{s_j,S_j} & M_{j+1}\end{pmatrix}.$$

Relation (2.10) and the induction hypothesis give the QR factorization of H_j

$$H_j = \left(\prod_{i=0}^{j-1} \mathcal{Q}^H_{(j+1-i),j}\right)\left(\prod_{i=1}^{j} \mathcal{F}_{i,j}\right)\begin{pmatrix} C_j \\ 0_{s,S_j}\end{pmatrix}.$$

We can write

$$\tilde{H}_{j+1} = \left(\prod_{i=0}^{j-1} \mathcal{Q}^H_{(j+1-i),(j+1)}\right)\left(\prod_{i=1}^{j} \mathcal{F}_{i,(j+1)}\right)\begin{pmatrix}\begin{pmatrix} C_j \\ 0_{s,S_j}\end{pmatrix} & \begin{pmatrix} n_{j+1,1} \\ n_{j+1,2}\end{pmatrix} \\ 0_{s_j,S_j} & M_{j+1}\end{pmatrix}.$$

where $\begin{pmatrix} n_{j+1,1} \\ n_{j+1,2}\end{pmatrix} = \left(\prod_{i=0}^{j-1} \mathcal{F}^H_{(j-i),j}\right)\left(\prod_{i=2}^{j+1} \mathcal{Q}_{(i),j}\right) N_{j+1}$.

Let F_{j+1} be the matrix of Householder transformation that triangularize $\begin{pmatrix} n_{j+1,2} \\ M_{j+1}\end{pmatrix}$, then we obtain the relation satisfied for $j+1$

$$\tilde{H}_{j+1} = \left(\prod_{i=0}^{j-1} \mathcal{Q}^H_{(j+1-i),(j+1)}\right)\left(\prod_{i=1}^{j+1} \mathcal{F}_{i,(j+1)}\right)\begin{pmatrix} C_{j+1} \\ 0_{s,S_{j+1}}\end{pmatrix}.$$

□

Proposition 3. *The following relations hold during the block GMRES with Arnoldi procedure.*

1. $R_0 = [\mathcal{V}_j, D_{j-1}, \tilde{V}_{j+1}]\left(\prod_{i=0}^{j-1} \mathcal{Q}^H_{(j-i),j}\right)\begin{pmatrix} \Pi_0 \\ 0_{S_j,s}\end{pmatrix}$.
2. $\|B - A(X_0 + \mathcal{V}_j Y)\|_F = \|\left(\prod_{i=0}^{j-1} \mathcal{F}^H_{(j-i),j}\right)\mathcal{Q}^H_{1,j}\begin{pmatrix} \Pi_0 \\ 0_{S_j,s}\end{pmatrix} - \begin{pmatrix} C_j \\ 0_{s,S_j}\end{pmatrix} Y\|_F$.
3. $Y_j = C_j \backslash E_j$.
4. $R_j = [\mathcal{V}_j, D_{j-1}, \tilde{V}_{j+1}]\left(\prod_{i=0}^{j-2} \mathcal{Q}^H_{(j-i),j)}\right)\left(\prod_{i=1}^{j} \mathcal{F}_{i,j}\right)\begin{pmatrix} 0_{S_j,s} \\ G_j\end{pmatrix}$.

Where Π_0 verifies the relation $R_0 = \tilde{V}_1 \Pi_0$, $Y \in \mathbb{C}^{S_j \times s}$, and

$$\begin{pmatrix} E_j \\ G_j\end{pmatrix} = \left(\prod_{i=0}^{j-1} \mathcal{F}^H_{(j-i),j}\right)\mathcal{Q}^H_{1,j}\begin{pmatrix} \Pi_0 \\ 0_{S_j,s}\end{pmatrix},$$

such that $E_j \in \mathbb{C}^{S_j \times s}$ and $G_j \in \mathbb{C}^{s \times s}$.

Proof. Proof is by induction for 1, and it is immediate for the rest. □

In the block GMRES method, a linear combination of the block residual could converge, while the system has not converged yet. This leads to unnecessary computations and memory loss. To remedy this issue, we use deflation technique based on detection of inexact breakdowns.

As explained in [59], Robbé and Sadkane introduced two criteria based on singular value decomposition to determine the convergent subspace. The first depends on the block residual. The second depends on the block basis vector. In a later paper [10], Calandra et al. reformulated the first criterion with a slight modification, leading to a different least squares problem.

The detection test is based on an SVD factorization of a matrix of size $(S_j + s) \times s$ at iteration j. This cost depends on the iteration number and it becomes expensive quickly. We propose in the next section a new strategy to reduce the problem to a matrix of size $s \times s$, hence the cost becomes independent of iteration. Moreover, we also study the detection of a test based on rank revealing QR.

2.4.2 Inexact breakdowns detection

Here we present the reformulation of the inexact breakdowns test that is presented in [59, 10]. This reformulation leads to a reduction of the dimension of the SVD test that is used to detect inexact breakdowns in the block residual of block GMRES. This theory can be applied to all block GMRES-like methods.

Proposition 4. *An SVD factorization on the matrix G_j is equivalent to an SVD factorization of $\bar{R}_j$.*

Proof. Proposition 3 proves that

$$\bar{R}_j = \left(\prod_{i=0}^{j-2} \mathcal{Q}_{(j-i),j}^H \right) \left(\prod_{i=1}^{j} \mathcal{F}_{i,j} \right) \begin{pmatrix} 0_{S_j,s} \\ G_j \end{pmatrix} \tag{2.21}$$

Let $G_j = U \Sigma W^H$ be the SVD factorization of G_j. Since $\left(\prod_{i=0}^{j-2} \mathcal{Q}_{(j-i),j}^H \right) \left(\prod_{i=1}^{j} \mathcal{F}_{i,j} \right)$ is unitary, we find that

$$\bar{R}_j = \left(\prod_{i=0}^{j-2} \mathcal{Q}_{(j-i),j}^H \right) \left(\prod_{i=1}^{j} \mathcal{F}_{i,j} \right) \begin{pmatrix} 0_{S_j,s} \\ U \end{pmatrix} \Sigma W^H$$

is an SVD factorization of $\bar{R}_j$ with $\left(\prod_{i=0}^{j-2} \mathcal{Q}_{(j-i),j}^H \right) \left(\prod_{i=1}^{j} \mathcal{F}_{i,j} \right) \begin{pmatrix} 0_{S_j,s} \\ U \end{pmatrix}$ standing for the left unitary factor. □

Corollary 1. *A rank revealing QR factorization on the matrix G_j is equivalent to a rank revealing QR factorization of $\bar{R}_j$.*

Proof. Proof is similar to the proof of Proposition 4 □

An important difference related to the reference test presented in [59, 10] is that the dimension of the factored matrix does not depend on the iteration number j. In the mentioned references this dimension is $S_j \times s$ at iteration j. Proposition 4 shows that this dimension is minimal.

In addition, it shows that there is no need to compute the residual $\bar{R}_j$ at each iteration to detect inexact breakdowns. Indeed to get the matrix G_j, it is sufficient to update E_j, G_j, by using F_j. This matrix, G_j is computed at each iteration in order to perform the stopping criterion. Thus, there is no need to solve the least squares problem entirely.

In the remaining of this section we introduce an inexact breakdowns detection based on rank revealing QR to reduce the cost of performing an SVD factorization.

We start from relation (2.9). Given the matrix $[D_{j-1}, \tilde{V}_{j+1}]$ and the matrix $G_j \in \mathbb{C}^{s\times s}$ that verifies the relation

$$\begin{pmatrix} 0_{S_j,s} \\ G_j \end{pmatrix} = \left(\prod_{i=0}^{j-1} \mathcal{F}^H_{(j-i),j}\right)\left(\prod_{i=2}^{j} \mathcal{Q}_{i,j}\right)\bar{R}_j$$

as presented in Proposition 3, find Q_{j+1} such that V_{j+1} spans the subspace related to the non convergent part of the block residual.

In [10] and [59], the authors propose a strategy based on the singular value decomposition of the matrix $\bar{R}_j \in \mathbb{C}^{(S_j+s)\times s}$. The detection test of the inexact breakdowns is done at every iteration. Hence, an SVD factorization of $\bar{R}_j \in \mathbb{C}^{(S_j+s)\times s}$ occurs at each iteration. During a cycle, the size of this problem grows linearly with the iteration number. We propose a new strategy to keep the dimension of the SVD problem constant and equals to $s \times s$. Furthermore, using rank revealing QR factorization [12] instead of SVD factorization reduces the computational complexity. Here, we derive the theory of that strategy.

Let ε_0 be a threshold given, and $G_j = S\mathcal{R}P^\top$ be a rank revealing QR factorization of the matrix $G_j \in \mathbb{C}^{s\times s}$. The matrix S stands for an orthonormal basis for the range of G_j, $\mathcal{R}$ is an upper triangular matrix and P is a permutation matrix. We can write the rank revealing QR relation in the form,

$$\begin{aligned} G_j &= \begin{pmatrix} S_+ & S_- \end{pmatrix}\begin{pmatrix} \mathcal{R}_1 & \mathcal{R}_2 \\ 0_{c_{j+1},s_{j+1}} & \mathcal{R}_3 \end{pmatrix}\begin{pmatrix} P_1^\top \\ P_2^\top \end{pmatrix} \\ &= S_+\begin{pmatrix} \mathcal{R}_1 & \mathcal{R}_2 \end{pmatrix}\begin{pmatrix} P_1^\top \\ P_2^\top \end{pmatrix} + S_-\mathcal{R}_3P_2^\top, \end{aligned} \tag{2.22}$$

with $\|\mathcal{R}_3\|_2 < \varepsilon_0$. Directly we have s_{j+1} is the numerical rank of G_j i.e., the number of columns in S_+.

To detect inexact breakdowns by using RRQR, the test depends on the same idea that is proposed in [59, 10]. The new basis vectors to be added should be related to the subspace that has not converged of the range of $\bar{R}_j$. We write the projection of the residual on the subspace perpendicular to $\mathcal{K}_j$ using the RRQR decomposition,

Note that $\mathcal{S}\mathcal{S}^H R_j = R_j$ where,

$$\mathcal{S} = [\mathcal{V}_j, D_{j-1}, \tilde{V}_{j+1}]\left(\prod_{i=0}^{j-2}\mathcal{Q}^H_{(j-i),j)}\right)\left(\prod_{i=1}^{j}\mathcal{F}_{i,j}\right)\begin{pmatrix}0_{S_j,s}\\ S\end{pmatrix}.$$

$$(I - \mathcal{V}_j\mathcal{V}_j^H)R_j = [0, D_{j-1}, \tilde{V}_{j+1}]\bar{R}_j \tag{2.23}$$

$$= [0, D_{j-1}, \tilde{V}_{j+1}]\left(\prod_{i=0}^{j-2}\mathcal{Q}^H_{(j-i),j)}\right)\left(\prod_{i=1}^{j}\mathcal{F}_{i,j}\right)\begin{pmatrix}0_{S_j,s}\\ S\end{pmatrix}\mathcal{R}P^\top \tag{2.24}$$

We want $Range(V_{j+1}) = Range((I - \mathcal{V}_j\mathcal{V}_j^H)\mathcal{S}_+\mathcal{S}_+^H R_j)$, where $\mathcal{S}_+$ is the first s_{j+1} columns of $\mathcal{S}$. Thus,

$$\begin{aligned} Range(V_{j+1}) &= Range\left([0, D_{j-1}, \tilde{V}_{j+1}]\left(\prod_{i=0}^{j-2}\mathcal{Q}^H_{(j-i),j)}\right)\left(\prod_{i=1}^{j}\mathcal{F}_{i,j}\right)\begin{pmatrix}0_{S_j,s}\\ S_+\end{pmatrix}\mathcal{R}P^\top\right)\\ &= Range\Big([D_{j-1}, \tilde{V}_{j+1}]\underline{S}\Big). \end{aligned} \tag{2.25}$$

Q_{j+1} is the unitary factor of the QR factorization of $\underline{S}$. As a result we have,

$$Range(V_{j+1}) \oplus Range(D_j) = Range(\tilde{V}_{j+1}) \oplus Range(D_{j-1}).$$

The columns of the matrix D_j form a subspace of the block Krylov subspace. They are chosen in the best way so that the new added basis vectors V_{j+1} are optimal. In fact, V_{j+1} helps to minimize only the largest singular values of the residual block in the next iteration. A threshold is given to separate the largest and the smallest singular values. Thus, the smallest singular values are neglected.

Algorithm 13 show how to compute the matrix Q_{j+1} (2.10).

Algorithm 13 Inexact breakdowns detection(G_j, ε)

Require: $G_j \in \mathbb{C}^{s\times s}$ and ε the tolerance of inexact breakdown.
Ensure: Q_{j+1} and s_{j+1}.
1: RRQR factorization of G_j, $G_j = S\mathcal{R}P^\top$.
2: $G_j = S_+(\mathcal{R}_1P_1^\top + \mathcal{R}_2P_2^\top) + S_-\mathcal{R}_3P_2^\top$. With $\mathcal{R}_3$ has maximum size with second norm less than ε. s_{j+1} is the rank of G_j.
3: QR factorization of $\underline{S}$ (2.25), Q_{j+1} is the unitary factor.

2.4.3 Deflation of eigenvalues

As the size of the memory is limited, we normally need to use the restart variant by disregarding all the built block Krylov subspace, and rebuilding a new one beginning

with the last residual. This means a loss of information. A common approach to keep useful information is to deflate small eigenvalues if their eigenvectors have converged or if they are well approximated by the end of the cycle [47, 20, 76, 30].

In the remaining of this section, we show how these eigenvalues and eigenvectors are chosen. We first recall a theorem from [20]. The algebraic formulation of the eigenvalues deflation preconditioner follows its result. We reformulated the theorem to make it conform with the context of this chapter.

Theorem 2. *Suppose that the matrix A is diagonalizable and let $\{\lambda_1, \cdots, \lambda_n\}$ be the set of eigenvalues of A with $|\lambda_1| \leq \cdots \leq |\lambda_n|$ and $\{u_1, \cdots, u_n\}$ be the corresponding normal eigenvectors. Given a threshold ε_1 set m to be the positive integer such that $|\lambda_m| < \varepsilon_1 \leq |\lambda_{m+1}|$. Let $U = (u_1, \cdots, u_m) = ZL$ be the QR factorization and $M = I_n + Z(\frac{1}{|\lambda_n|} Z^H A Z - I_m) Z^H$, then*

1. *The matrix M is invertible and its inverse $\tilde{M} = I_n + Z(|\lambda_n|(Z^H A Z)^{-1} - I_m) Z^H$.*

2. *The matrix AM^{-1} has eigenvalues $\{\lambda_n, \cdots, \lambda_n, \lambda_{m+1}, \cdots, \lambda_n\}$ with $\{u_1, \cdots, u_n\}$ as corresponding eigenvectors.*

Proof. A similar proof using invariant subspaces is given in [20]. □

Well approximated eigenvalues and eigenvectors

Let S_1 be the upper $k_{cycle} \times k_{cycle}$ sub-matrix of the matrix H_{cycle}, where $cycle$ is the number of last iteration in block GMRES. Suppose that S_1 is diagonalizable (it is sufficient to suppose that so is A), and let $\{\lambda_1, \cdots, \lambda_m\}$ be the eigenvalues of S_1 with absolute value less than a given threshold, ε_1. In Algorithm 14 we propose an approach to measure the approximated eigenvector residual norm. No multiplication by the matrix A is necessary. Actually, we just need the matrix H_{cycle} and the eigenvector u in the block Krylov subspace to perform the test. The following theorem addresses the theoretical part that Algorithm 14 depends on.

Proposition 5. *Let H_{cycle} be the matrix verifying*

$$A\mathcal{V}_{cycle} = [\mathcal{V}_{cycle+1}, \mathcal{D}_{cycle}] H_{cycle}$$

and denote by S_1 the maximal square submatrix of H_{cycle} obtained by deleting lines from the bottom of H_{cycle}, such that $H_{cycle} = \begin{pmatrix} S_1 \\ S_2 \end{pmatrix}$. Let u be an eigenvector of the matrix S_1 with eigenvalue λ. Then

$$\|A\mathcal{V}_{cycle} u - \lambda \mathcal{V}_{cycle} u\|_2 = \|S_2 u\|_2.$$

Proof.

$$\begin{aligned} A\mathcal{V}_{cycle}u &= [\mathcal{V}_{cycle+1}, D_{cycle}]H_{cycle}u \\ &= [\mathcal{V}_{cycle+1}, D_{cycle}]\begin{pmatrix} S_1 \\ S_2 \end{pmatrix}u \\ &= \mathcal{V}_{cycle}S_1u + [V_{cycle+1}, D_{cycle}]S_2u \\ &= \lambda\mathcal{V}_{cycle}u + [V_{cycle+1}, D_{cycle}]S_2u. \end{aligned}$$

Since $[V_{cycle+1}, D_{cycle}]$ is unitary, it yields

$$\|A\mathcal{V}_{cycle}u - \lambda\mathcal{V}_{cycle}u\|_2 = \|S_2u\|_2.$$

□

Scaling the spectrum of the linear system is taken into account in practice. To decide if a vector $\mathcal{V}_{cycle-1}u$ is a good approximation of an eigenvector, we compute $\frac{\|S_2u\|_2}{|\lambda_{max}|}$. If it is less than the given threshold, this vector will be deflated. A good approximation of $|\lambda_{max}|$ is computed after the first restart, since this eigenvalue converges fast.

2.4.4 RD-BGMRES(m)

In Algorithm 15 we present the *Restarted Deflated BGMRES(m)*, where m is the maximum number of vectors to be saved in memory, including both basis vectors and approximated eigenvectors. This algorithm is the main part of the CPR-EGMRES method. It is applied at each iteration of the CPR solver as presented in the following section. We note that the parameters $\varepsilon_0, \varepsilon_1$ in Algorithm 15 are not related, see Table 2.3 and the discussion that follows it in Section 2.6.

2.5 CPR-EGMRES

In this section, we introduce the constrained pressure residual preconditioner with EGMRES. This preconditioner was first introduced by [80] as a preconditioner for the solution of systems of linear equations arising from the simulations of reservoirs. In simulations of reservoirs, the overall system is of mixed character. However, the pressure field usually has a near elliptic behavior with long range coupling, while the remaining equations (referred to as saturation equations) often possess near hyperbolic character with steep local gradients [63, 80]. As a direct consequence, the linear systems in simulations of reservoirs are a natural target for a two-stage preconditioning strategy.

Algorithm 14 Deflation of eigenvalues($A, \mathcal{V}_{cycle}, H, Z, |\lambda_{max}|, \varepsilon, nev$)

Require: The matrix A, the basis of the block Krylov subspace $\mathcal{V}_{cycle}$, the matrix $H = \begin{pmatrix} S_1 \\ S_2 \end{pmatrix}$, threshold for convergence of eigenvalues and eigenvectors ε, the maximum number of eigenvalues to deflate nev, already deflated eigenvectors Z (optional), $|\lambda_{max}|$ (optional).

Ensure: nev, and the terms used in the preconditioner Z and $Z^H AZ$, an approximation of the magnitude of the largest eigenvalue $|\lambda_{max}|$ (optional).

1: Calculate the eigenvalues $\{\lambda_1, \cdots, \lambda_p\}$ of S_1 with absolute value less than ε and their corresponding normal eigenvectors $\{u_1, \cdots, u_p\}$.
2: Set $U = [\,]$.
3: **if** $|\lambda_{max}|$ is not provided **then**
4: Compute $|\lambda_{max}|$.
5: **end if**
6: **for** $i = 1 : min\{p, nev\}$ **do**
7: **if** $\|S_2 u_i\|_2 < |\lambda_{max}|\varepsilon$ **then**
8: $U = [U, u]$.
9: $nev = nev - 1$.
10: **end if**
11: **end for**
12: QR factorization of U, $U = ZL$.
13: Expand the vectors of Z to $\mathbb{C}^n$, $Z = \mathcal{V}_{cycle} Z$.
14: Form $Z^H AZ = L^H \Lambda L$ to use in the preconditioner, Λ is a diagonal matrix with the deflated λ_i.

Algorithm 15 RD-BGMRES(m)

Require: The matrix A, the right-hand side B, the initial guess X_0, the tolerance of convergence ε_0, the tolerance of eigenvalues and eigenvector residual norm ε_1, the maximal number of deflated eigenvalues nev_{max}, the maximal number of cycles $cycle_{max}$, the maximal number of vectors to be saved in memory m, preconditioner M (I if not given).

Ensure: The approximate solution X_a of the system $AX = B$, the preconditioner M.

Set $nev = nev_{max}$, $R_0 = B - AX_0$.
for $cycle = 1 : cycle_{max}$ **do**
  if $cycle > 1$ and $nev > 0$ **then**
    Call the algorithm (14) to obtain Z.
    Update the preconditioner M^{-1} and update nev.
  end if
  QR factorization of R_0, $R_0 = \tilde{V}_1 \Pi_0$.
  Call Algorithm 13(Π_0, ε) to determine the matrix Q_1 and s_1. $S_1 = s_1$.
  Set $E_0 = Q_1 \Pi_0$, $G_0 = 0_{s_1,s}$
  $[V_1, D_0] = \tilde{V}_1 Q_1$, with $V_1 \in \mathbb{C}^{n \times s_1}$ and $D_0 \in \mathbb{C}^{n \times s - s_1}$. Set $j = 0$.
  while $S_{j+1} + s < m$ **do**
    Set $j = j + 1$. $W = AM^{-1}V_j$.
    Orthogonalize W against $\mathcal{V}_j$ and D_{j-1}.
    QR factor W. Build $\tilde{V}_{j+1}$ and get $\mathcal{F}_{j,j}$ to update the QR factorization of $\tilde{H}_j$ as in (2.20).
    $$\begin{pmatrix} E_j \\ G_j \end{pmatrix} = \mathcal{F}_{j,j}^H \begin{pmatrix} E_{j-1} \\ G_{j-1} \\ 0_{s_j,s} \end{pmatrix}.$$
    Call the algorithm (13)(G_j, ε_1) to determine the matrix $\mathcal{Q}_{j+1,j}$ and s_{j+1}.
    $S_{j+1} = S_j + s_{j+1}$.
    if $\|G_j\|_F < \varepsilon_0$ **then**
      Break.
    end if
  end while
  $Y_j = C_j \setminus E_j$. $X_j = M^{-1}\mathcal{V}_j Y_j$. $X_a = X_0 + X_j$.
  $R_j = R_0 - AX_j$. Set $R_0 = R_j$, and $X_0 = X_a$.
end for

2.5.1 Two-stage preconditioning

The two-stage preconditioning formula is given by

$$M_{1,2}^{-1} = M_2^{-1}[I - AM_1^{-1}] + M_1^{-1}.$$

M_2 is a preconditioner for the second level or stage, whereas, M_1 preconditions the first level. In simulations of reservoirs, the first level is related to the pressure system. The second level is related to the whole system. The CPR preconditioner satisfies

$$M_{CPR}^{-1} = M^{-1}[I - AC(W^T AC)^{-1}W^T] + C(W^T AC)^{-1}W^T.$$

where C is an $(n_{eqn} \cdot n_{cell})$ by n_{cell} block diagonal matrix (n_{eqn} is the number of unknowns per cell and n_{cell} is the total number of cells in the model). As pressure is the last unknown in each cell, C is given by

$$C = \begin{bmatrix} e_p & & & \\ & e_p & & \\ & & \ddots & \\ & & & e_p \end{bmatrix}.$$

$e_p = [0, \ldots, 0, 1]^T$, and W^T is an n_{cell} by $(n_{eqn} \cdot n_{cell})$ block diagonal matrix. A choice for W^T is $W^T = C^T.J^{-1}$ where J is a block Jacobi preconditioner. If we see the formula of the CPR preconditioner, we remark that a solution of the pressure system is needed in the application of the preconditioner. The authors in [63] propose the use of a multigrid solver which is efficient in terms of iteration count but lacks scalability for the set up time. As the application of the preconditioner occurs every iteration, we need to solve a linear system of equations corresponding to the pressure matrix (the same in all iterations) every iteration. For the second level we propose the usage of our method EGMRES in the mode of restart and deflation. The procedure of the first level is explained as following. We follow the notations of the CPR preconditioner and let B be a right-hand side. The system to be solved is $AX = B$. Construct $P(B)$, the enlarged residual and normalize it (vectors of $P(B)$ are already orthogonal). Let V_1 be the result.

At the first iteration of the second level, the application of the preconditioner takes effect on the first block of basis vectors V_1. We are going to explain how to compute

$$M_1^{-1}V_1 = C(W^T AC)^{-1}W^T V_1.$$

This application is performed by solving the system

$$(W^T AC)X = W^T V_1$$

by using RD-EGMRES and then extending X to the second level by C.
$W^T V_1$ restricts the enlarged residual to the pressure level. Following the definition of the enlarged residual Section 2.3, $W^T V_1$ also has the form of an enlarged residual.

Computing

$$(W^T AC)^{-1} W^T V_1$$

is performed as an approximation of the solution. It is obtained by RD-EGMRES. We can write this system on the form

$$(W^T AC)X = W^T V_1.$$

Solving this first level by using RD-EGMRES is natural since as mentioned the right-hand side $W^T V_1$ has the enlarged form. Once we obtain the approximate solution X, we extend it to the second level by multiplying it with C. We save in memory the deflation preconditioner M_{def} to use it in next iterations. Then, we continue the rest of the iteration on the second level. To reapply the first level preconditioner in next iterations, we benefit from the deflation of eigenvalues that we obtained in the first iteration M_{def}. Numerical experiments in Section 2.6 show that a very fast convergence is achieved on both levels.

We note that it is possible to use EGMRES method only on the pressure level. Whereas, GMRES (or FGMRES) is used on the saturation level. In application, we consider EGMRES as a flexible preconditioner on the pressure level. Thus, the stopping criterion can be a fixed number of iterations or a large threshold for convergence. The method obtained is more flexible and efficient as the numerical tests presented in Section 2.6 will show.

2.6 Numerical experiments

In this section, **RD-EGMRES** stands for EGMRES with restart, deflation of eigenvalues and inexact breakdowns detection. RD-GMRES refers to restarted GMRES with deflation of eigenvalues. Here we investigate the numerical behavior of EGMRES. We compare it to GMRES and BGMRES.

2.6.1 Test problems

Our test matrices arise from the discretization of four types of challenging problems: simulations of reservoirs, seismic imaging, linear elasticity and diffusion problems [29, 1, 55]. All numerical experiments are done by using Matlab 2016R. If it is not otherwise specified, the results correspond to RD-EGMRES.

The matrices SKY3D and ANI3D arise from boundary value problem of the diffusion equation on the (3-D) unit cube Ω:

$$-\mathrm{div}(\kappa(x)\nabla u) = f \qquad \text{in } \Omega, \tag{2.26}$$

$$u = 0 \qquad \text{on } \Gamma_D, \tag{2.27}$$

$$\frac{\partial u}{\partial n} = 0 \qquad \text{on } \Gamma_N, \tag{2.28}$$

The tensor κ is a given coefficient of the partial differential operator, $\Gamma_D = [0,1] \times \{0,1\} \times [0,1]$, Γ_N is chosen as $\Gamma_N = \partial\Omega \setminus \Gamma_D$ and n denotes the exterior normal vector to the boundary of Ω. The matrix ANI3D is obtained by considering anisotropic layers: the domain is made of 10 anisotropic layers with jumps of up to four order of magnitude and an anisotropy ratio of 10^3 in each layer. Those layers are parallel to $z = 0$, of size 0.1, and inside them the coefficients are constant: $\kappa_y = 10\kappa_x$, $\kappa_z = 100\kappa_x$. This problem is 3D, discretized on a Cartesian grid of size $20 \times 20 \times 20$. The Elasticity3D100 matrix arise from the linear elasticity problem with Dirichlet and Neumann boundary conditions defined as follows

$$\operatorname{div}(\sigma(u)) + f = 0 \qquad \text{in } \Omega, \tag{2.29}$$

$$u = 0 \qquad \text{on } \Gamma_D, \tag{2.30}$$

$$\sigma(u) \cdot n = 0 \qquad \text{on } \Gamma_N. \tag{2.31}$$

Ω is a unit square (2D) or a unit cube (3D). The matrices Elasticity3D100 and Elasticity2D125 correspond to this equation discretized using a triangular mesh with $100 \times 10 \times 10$ vertices for the (3D) case and 125×10 vertices for the (2D) case. Γ_D is the Dirichlet boundary, Γ_N is the Neumann boundary, f is a force, u is the unknown displacement field. The Cauchy stress tensor $\sigma(.)$ is given by Hooke's law: it can be expressed in terms of Young's Modulus E and Poisson's ratio ν. n denotes the exterior normal vector to the boundary of Ω. For a more detailed description of the problem see [50] and [30]. We consider discontinuous E and ν: $(E_1, \nu_1) = (2 \times 10^{11}, 0.25)$ and discontinuous E in (2D): $(E_1, \nu_1) = (10^{12}, 0.45)$ and $(E_2, \nu_2) = (2 \times 10^6, 0.45)$. The matrices BIGCO24 and BIGP1 are obtained from our in-house prototype code at Total, that simulates a complex enhanced oil recovery (EOR) mechanism. This simulator relies on a finite volume discretization and a two-point flux approximation. BIGP1 comes from the simulation of water injection using a black-oil model. The permeability field is heterogeneous (sector model from a real field case). The grid has 42332 active cells. BIGCO24 corresponds to a simulation of water and gaz injection using a compositional model (8 hydrocarbon components). The permeability field is heterogeneous. The grid has 83587 active cells.

In Table 2.1 we present our test matrices. The linear systems arising from simulations of reservoirs and linear elasticity have one right-hand side. Seismic imaging systems have multiple right-hand sides. We use a block Jacobi preconditioner with 128 blocks in all our experiments. The threshold of convergence in all our tests is 10^{-8}.

Table 2.2 shows a brief comparison between EGMRES and GMRES for several matrices in our set. The number of iterations decreases drastically by increasing the enlarging factor of the Krylov subspace. EGMRES and GMRES use the same number of communication messages per iteration. Thus, an overall communication reduction is accomplished by EGMRES. For example, an enlarging factor $EF = 32$ reduces the iteration count by a factor of 12 with the matrix Elasticity2D125. Figures 2.2 and 2.3 show that using $RRQR$ or SVD tests to detect the inexact breakdowns does not affect the robustness of the method. On the contrary, they keep the efficiency of the method and they reduce

Matrix name	Type	N	NnZ	Real	HPD	κ
BIGCO24	Saturation	752283	5495556	yes	no	2×10^{11}
P-BIGCO24	Pressure	83587	539605	yes	no	10^9
BIGP1	Saturation	169328	2469485	yes	no	4×10^{13}
P-BIGP1	Pressure	42332	275946	yes	no	10^8
Seismic1	Seismic imaging	11285	55380	no	no	9×10^3
Seismic2	Seismic imaging	69611	345450	no	no	6×10^4
Seismic3	Seismic imaging	123414	613600	no	no	10^5
Elasticity3D100	Elasticity	36663	1231497	yes	yes	3×10^7
Elasticity2D125	Elasticity	31752	378000	yes	yes	10^8
SKY3D	Skyscraper	8000	53000	yes	yes	10^5
ANI3D	Anisotropic Layers	8000	53600	yes	yes	10^3

Table 2.1: Matrices used for tests. N is the size of the matrix, NnZ is the number of nonzero elements. HPD stands for Hermitian Positive Definite. κ is the condition number related to the second norm.

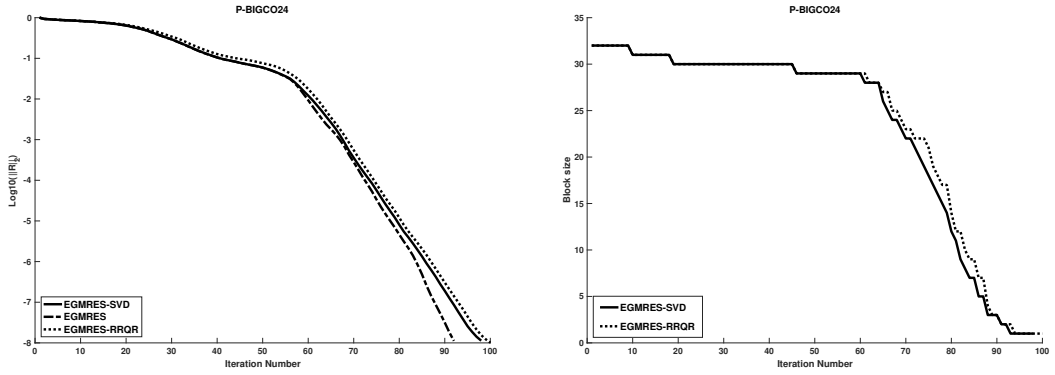

Figure 2.2: On the left, the convergence of EGMRES with RRQR and SVD strategies to reduce the size of block vector. On the right, impact of inexact breakdowns detection on the block vector size by using each strategy.

Matrix	It_G	EF	It_{EG}	Dim_{EK}	It_{EG-SVD}	Dim_{EK}	$It_{EG-RRQR}$	Dim_{EK}
P-BIGCO	319	4	212	848	218	747	218	748
		8	152	1216	158	1124	156	1127
		16	114	1824	125	1622	126	1636
		32	92	2944	98	2313	100	2360
P-BIGP1	822	4	426	1704	435	1601	434	1605
		8	290	2320	294	2176	293	2195
		16	198	3168	201	2929	203	2965
		32	133	4256	139	3856	141	3912
Elasticity3D100	599	4	213	852	213	754	235	779
		8	142	1136	144	945	177	988
		16	101	1616	105	1296	105	1321
		32	78	2496	80	1858	80	1913
Elasticity2D125	1572	4	489	1956	491	1854	490	1858
		8	287	2296	291	2155	289	2168
		16	188	3008	191	2659	190	2682
		32	133	4256	136	3520	136	3566
ANI3D	94	4	85	340	85	314	86	320
		8	80	640	82	589	82	599
		16	75	1200	77	1093	77	1110
		32	65	2080	66	1901	67	1933
SKY3D	376	4	220	880	221	842	233	863
		8	125	1000	125	976	128	981
		16	73	1168	73	1102	73	1109
		32	46	1472	46	1336	46	1347

Table 2.2: Comparison between inexact breakdowns detection methods. It_{Method} stands for the number of iterations to achieve convergence. *G*: GMRES as standard method. *EG*: EGMRES without inexact breakdowns detection. *EG-SVD*: EGMRES with inexact breakdowns detection using the SVD, as presented in [10], and *EG-RRQR*: EGMRES with inexact breakdowns detection using rank revealing test (see Section 2.4.2). *EF* is the enlarging factor. Preconditioner: 128 block Jacobi. Threshold of convergence is 10^{-8}.

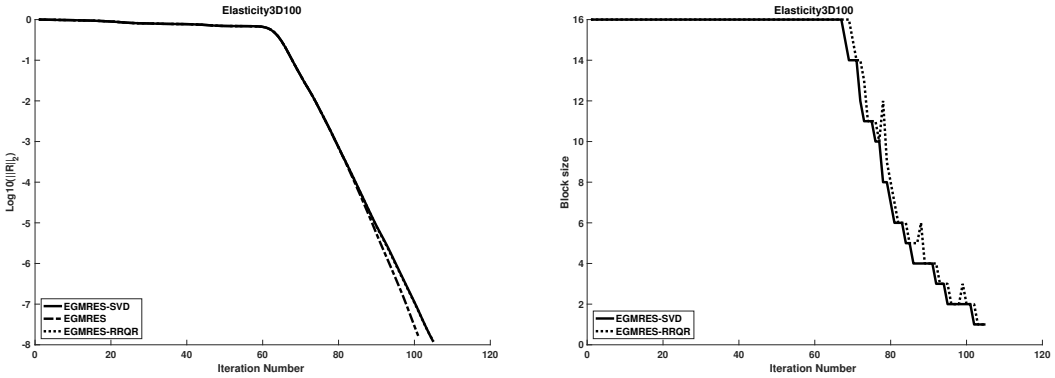

Figure 2.3: On the left, the convergence of EGMRES with RRQR and SVD strategies to reduce the size of block vector. On the right, impact of inexact breakdowns detection on the block vector size by using each strategy.

both the memory and the computational costs. A gain in the number of iterations up to 82% with pressure system P-BIGP1 ($EF = 32$) and with Elasticity3D100 ($EF = 16$) is obtained. We notice also in Figure 2.2 that starting from the seventh iteration, the size of block vectors that are added to the basis begins to decrease with both strategies, RRQR and SVD. Starting with a block of size 32, EGMRES ends up by adding 3 vectors while maintaining the rate of convergence as if 32 vectors were added at each iteration.

Table 2.3 outlines the effect of deflating eigenvalues and the accuracy of estimating eigenvectors on the convergence of EGMRES. It shows results for RD-EGMRES(m) with two values of eigenvector convergence threshold. The value of m varies in the set $\{250, 500\}$. When the maximal number of vectors to be saved is relatively small, choosing a relatively small EF, with a threshold μ (threshold for the criteria of eigenvalues deflation Algorithm 14) of order 10^{-2} leads to a fast convergence and maintains the speed of the convergence as if no restart is done. For example, the challenging matrix P-BIGP1 with $EF = 4$. EGMRES without restart needs 434 iterations to achieve convergence, while RD-EGMRES(250) needs 496 iterations. Comparing to GMRES that iterates 822 times with no restart, this difference is small. For our experiments, using a threshold $\mu = 10^{-2}$ is efficient in most cases. Choosing a larger threshold leads to a larger number of eigenvectors being deflated. This yields to less iterations per cycle that are not enough to reach convergence fast. A smaller threshold induces deflating a small number of eigenvectors. Thus, we observe a stagnation of the residual. This results in slow convergence. The matrix ANI3D with 128 block Jacobi preconditioner has a small condition number, $\kappa = 232.6$. Thus, the impact of using EGMRES with such system is not important. In most cases, comparing to the full method where no restart is done, RD-EGMRES keeps the rate of convergence. For some cases, when the enlarging factor is big and the maximal basis size is small, the method fails to converge. This occurs since the number of iterations per cycle is very few. For example, the matrix Elasticity2D125 with $EF = 32$ and a maximal basis size of 250 vectors, the method does not converge.

Matrix	EF	It_{EG250}				It_{EG500}			
		μ_1	$DimDef$	μ_2	$DimDef$	μ_1	$DimDef$	μ_2	$DimDef$
P-BIGCO	1	321	35	322	15	319	0	319	0
	4	222	87	225	34	221	69	220	28
	8	169	145	179	42	161	122	167	43
	16	157	204	182	49	136	193	149	55
	32	+	+	419	55	117	266	160	56
P-BIGP1	1	835	114	835	42	825	72	826	31
	4	512	213	496	81	451	224	450	81
	8	+	+	399	103	331	302	332	105
	16	+	+	1268	137	284	403	277	141
	32	+	+	+	+	+	+	451	166
SKY3D	1	406	22	406	22	376	0	376	0
	4	310	85	307	81	268	86	256	85
	8	193	105	206	96	158	85	159	82
	16	139	119	166	101	102	101	105	96
	32	+	+	+	+	75	102	74	96
ANI3D	1	94	0	94	0	94	0	94	0
	4	86	30	87	1	85	0	85	0
	8	84	52	88	1	82	60	84	1
	16	83	118	100	1	79	103	88	1
	32	108	186	105	2	74	198	89	2
Elasticity2D125	1	1712	224	1689	90	1620	229	1621	85
	4	+	+	650	96	532	256	537	96
	8	+	+	518	109	336	296	356	96
	16	+	+	2245	150	256	357	338	123
	32	+	+	+	+	319	447	394	142
Elasticity3D100	1	608	65	658	29	602	71	632	30
	4	221	82	267	33	216	68	235	28
	8	169	108	220	36	150	104	191	32
	16	159	140	245	40	113	133	160	33
	32	304	186	379	50	99	165	180	38

Table 2.3: Comparison between two tolerance values of residual eigenvectors, $\mu_1 = 5 \times 10^{-2}$, $\mu_2 = 10^{-2}$. It_{Method} stands for number of iterations using *Method* as algorithm, $EG(m)$ for RD-EGMRES where m is the maximum number of stored vectors either deflated or basis vectors. Preconditioner: 128 block Jacobi, '+' means that a stagnation of the norm of the residual occurs and no convergence is achieved. Threshold of convergence is 10^{-8}.

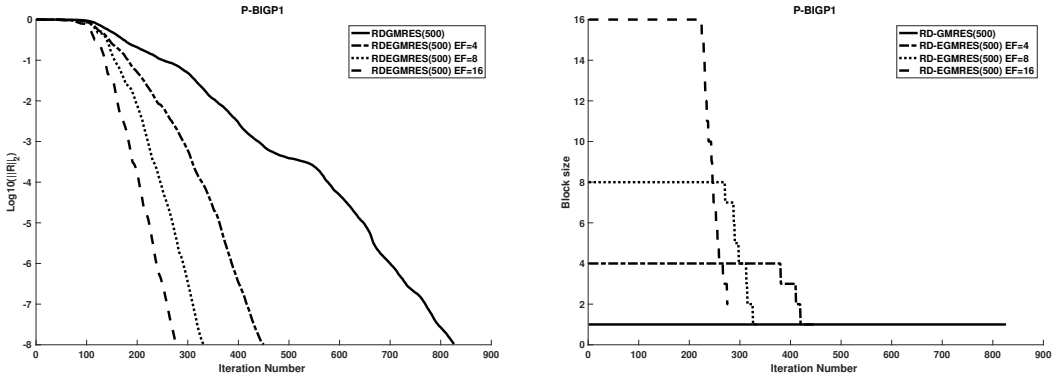

Figure 2.4: On the left, RD-EGMRES convergence with different enlarging factors. On the right, impact of inexact breakdowns detection, based on RRQR criterion, on the size of the block vectors.

Indeed, RD-EGMRES(250) does less than 7 iterations per cycle. That was not enough to maintain the efficiency. However, with the same basis size and an $EF = 8$ a gain of factor 3.2 is obtained.

2.6.2 EGMRES and RD-EGMRES

Here we present numerical tests for EGMRES and RD-EGMRES on the set of matrices presented in Table 2.1. In Figure 2.4, we show the impact of enlarging the Krylov subspace on the number of iterations to reach convergence. Although the maximal dimension of the search subspace is fixed, increasing the enlarging factor decreases the number of iterations. This efficiency is due to the richness of the enlarged Krylov subspace and the deflation of eigenvalues (see Figures 2.5 and 2.6). The number of iterations is reduced by a factor of 3 with $EF = 16$. Furthermore, we also display the impact of inexact breakdowns detection on the size of the block vectors. Up to $EF = 16$, RD-EGMRES requires approximately the same number of iterations as the full method (EGMRES) needs to reach convergence. However, the cost of orthogonalization is reduced drastically. Figures 2.5 and 2.6 show the efficiency of RD-EGMRES to deflate eigenvalues along cycles. We compute the 25^{th} smallest eigenvalues of the original system as a reference. We run RD-GMRES and RD-EGMRES and compare their ability to deflate eigenvalues. Deflated eigenvalues are shifted such that the spectrum of the deflated system becomes more clustered. In the figure this appears as a translation to the top. We run RD-EGMRES(250) with different enlarging factors $EF = 4$ or 8. At the end of each cycle, we compute the 25^{th} smallest eigenvalues of the deflated system. We do the same for deflated and restarted GMRES with the same basis size 250. We compare with the reference. RD-EGMRES(250) deflates better than the restarted and deflated GMRES. On the right of (Figure 2.5) RD-EGMRES(250) with $EF = 4$ deflate eigenvalues after 3 cycles better than RD-GMRES(250) after 3 cycles in which it reaches convergnece. The first 3 cycles of RD-EGMRES(250) with $EF = 4$ perform sparse-matrix applications

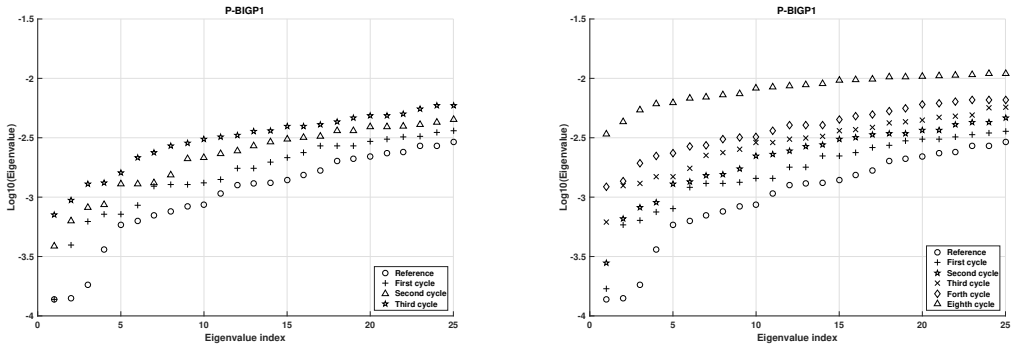

Figure 2.5: Eigenvalues deflation for P-BIGP1. On the left $EF = 1$, on the right $EF = 4$. Preconditioner: 128 block Jacobi.

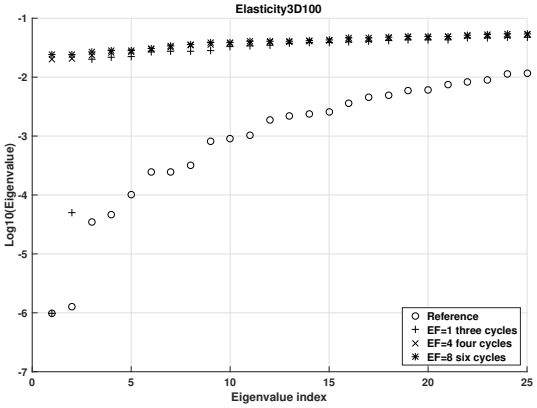

Figure 2.6: Comparison: eigenvalues deflation over cycles with different enlarging factors. Eigenvalues deflation on Elasticity3D100. Preconditioner: 128 block Jacobi.

less than the number of these type of operations that RD-GMRES(250) performs in the first cycle. In Figure 2.6 the test matrix, Elasticity3D100, preconditioned by 128 block Jacobi, has a condition number $\kappa = 2 \times 10^6$. RD-GMRES(250) converges after 3 cycles without deflating the smallest eigenvalue. RD-EGMRES(250) with $EF = 4$ and $EF = 8$ deflate all eigenvalues less than the threshold chosen 10^{-2} after 4 and 6 cycles respectively.

Table 2.4 shows results for seismic imaging problems with N_p right-hand sides. Enlarging the block Krylov subspace results in faster convergence. In our set of seismic systems, we observe that the worse the system is conditioned the more the gain is obtained. Seismic systems 1, 2 and 3 have condition numbers 9×10^3, 6×10^4 and 10^5 respectively. Nevertheless, the gains obtained by RD-EGMRES(500) (comparing to Restated and Deflated Block GMRES) are 45%, 53% and 58% respectively.

Matrix	N_p	It_{BGS}	ND	It_{EG}
Seismic1	4	468	4	260
			8	326
Seismic2	4	587	4	320
			8	275
Seismic3	4	596	4	309
			8	251

Table 2.4: Comparison between RD-EGMRES and RD-BGMRES. Threshold of convergence is 10^{-8}. The maximal size of the search subspace is 500 including the deflated eigenvectors.

Matrix	EF	$It_{EG}B_1$	N_{ev}	$It_{EG}B_2$
P-BIGCO24	1	319	0	319
	4	220	28	125
	8	167	43	100
	16	149	55	94
P-BIGP1	1	826	31	431
	4	450	81	179
	8	332	105	145
	16	277	141	149

Table 2.5: Influence of the enlarging factor of the Krylov subspace with multiple right-hand sides, each given at a time. N_{ev} is the number of eigenvectors deflated after the solution of $Ax = B_1$. Threshold of convergence is 10^{-8}. The maximal size of the search subspace is 500 including the deflated eigenvectors.

2.6.3 CPR-EGMRES

In the following we show results for EGMRES in the context of Constrained Pressure Residual preconditioner. Simulation of reservoirs induce linear systems that have a coupling structure. The pressure system, appearing as a sub-matrix, and the global system standing for the saturation system. The CPR-Preconditioning technique [63] is widely used for such problems. The main operation while solving the saturation system, by using CPR-Preconditioner, is to solve a pressure system at each iteration. For this reason, more results on pressure systems are presented rather than other problems. To view the efficiency of using RD-EGMRES to solve the saturation systems, Table 2.5 presents results for the following type of test: solve the pressure system $AX = B_1$ using RD-EGMRES and save in memory the preconditioner M^{-1} Algorithm 14 constructed during the solution. Then, solve the pressure system $AX = B_2$ using RD-EGMRES preconditioned by M^{-1}. In Table 2.5, to solve $AX = B_2$, RD-GMRES iterates

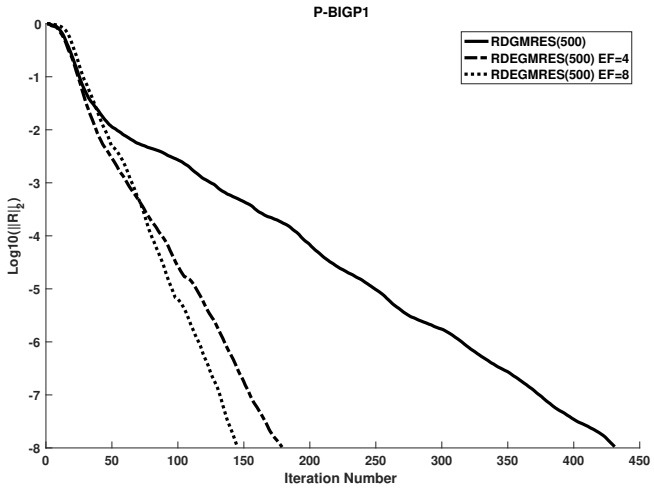

Figure 2.7: Deflation of linear system of equations with multiple right-hand sides each given at a time. Results show the impact of the enlarging factor on the convergence of $AX = B_2$ and a comparison with RD-GMRES(500). $AX = B_1$ was previously solved by using RD-GMRES(500) and a deflation preconditioner is constructed to solve $AX = B_2$.

more than what RD-EGMRES needs to solve $AX = B_1$. This is very important for CPR-Preconditioning. Indeed, every application of the CPR preconditioner requires the solution of the first level (Section 2.5). This yields to a linear system of equations with multiple right-hand sides each given at the application of the preconditioner. Figure 2.7 illustrates the impact of deflating eigenvalues by using RD-EGMRES. It is true that for the factor $EF = 16$, RD-EGMRES iterates approximately the same as for the factor $EF = 8$. Smallest eigenvalues have been deflated for both cases, such that improving further the conditioning is not useful. However, no stagnation of the residual occurs, in contrast to RD-GMRES. That is related to the comparison between RD-GMRES and RD-EGMRES about the deflation of smallest eigenvalues. Two practical approaches for using the CPR-EGMRES preconditioner, precisely on the pressure level, are proposed here:

- using a stopping criterion as the norm of the residual, such that it has to be the same as the one for the saturation system,
- using a specified number of iterations[1].

In the first type, we do not need to use a flexible form while in the second it is necessary to use the flexible variant. Table 2.6 shows numerical results using these two approaches. For the results of GMRES in Table 2.6, we use CPR preconditioner by using a direct

[1] The first iteration on the second level, RD-EGMRES performs sufficient iterations on the first level to get deflation information.

Matrix	*GMRES*		*EF*	*FEGMRES*			*EGMRES*	
	It_G	*RelErr*		it_P	it_{FEG}	*RelErr*	It_{EG}	*RelErr*
BIGCO24	20	7×10^{-8}	4	5	31	2×10^{-9}	14	8×10^{-6}
			4	10	19	8×10^{-11}		
			8	5	20	2×10^{-9}	12	9×10^{-6}
			8	10	15	3×10^{-10}		
			16	5	17	9×10^{-10}	10	9×10^{-6}
			16	10	14	9×10^{-11}		
BIGP1	81	5×10^{-10}	4	5	78	7×10^{-11}	79	2×10^{-9}
			4	10	68	3×10^{-10}		
			8	5	62	10^{-10}	57	2×10^{-10}
			8	10	59	6×10^{-11}		
			16	5	54	4×10^{-11}	46	2×10^{-9}
			16	10	52	5×10^{-11}		

Table 2.6: CPR-EGMRES. it_G refers to the number of iterations of GMRES by using a direct solver on the pressure level. it_P stands for the fixed number of iterations, being done by RD-EGMRES, in the pressure level. it_{EG} refers to the number of iterations of EGMRES to reach convergence, it uses RD-EGMRES as a solver for the pressure level. *RelErr* refers to the relative error in the solution. FEGMRES stands for Flexible EGMRES. Threshold of convergence is 10^{-8}.

LU solver in the level of pressure. This is the theoretical CPR approach (Section 2.5). In our experiments we compare two fixed number of iterations, 5 and 10, with the second approach. RD-EGMRES with the CPR preconditioner used for solving the two levels of the system converges faster than the previously described CPR-GMRES solver. Both previously mentioned approaches result in less number of iterations to reach convergence than the CPR-GMRES solver where the sub-system of pressure is solved by using a direct LU solver. We obtain a gain of up to 50% for the non-flexible variant with $EF = 16$ and up to 35% for the flexible variant with $EF = 16$, when solving systems with BIGCO24 and BIGP1 respectively. We have to mention that using the standard GMRES, either in block or simple case, on the saturation level without the flexible variant, causes a stagnation of the real residual norm. Whereas the norm of the residual in the (enlarged) Krylov subspace still decreases. That explains why the error in the solution in Table 2.6 for non-flexible methods is far from the error related to the flexible variant.

2.7 Conclusion

We introduced Enlarged GMRES, a linear solver based on two previous works, GMRES [61], and a communication reducing approach, enlarged Krylov subspace [29]. At each iteration of Enlarged GMRES, we add multiple basis vectors for each right hand-side,

while keeping the same number of messages required for computing this method in parallel. This results in faster convergence. Due to limited memory and the arithmetic cost of orthogonalization, restarting the method is necessary. One solution to this problem is the reduction of the added block vectors, once they are not useful. It maintains the rate of convergence of the method, as if no reduction is done, and decreases the computational and memory cost. Starting from the theory of exact and inexact breakdowns introduced in [59], we developed a new theoretical and practical strategy to detect inexact breakdowns based on rank revealing QR of a $T \times T$ matrix where T is the number of columns of the initial enlarged block residual. This strategy, Algorithm 13, is used to reduce the size of the block vectors. It can be applied for all block GMRES-like methods. The necessity to solve linear systems of equations with multiple right-hand sides, each given at a time, prompted us to use deflation of eigenvalues to maintain the rate of convergence when a restart occurs. To this end, we used Proposition 5, originally presented in [20]. This theorem gives an algebraic formulation for a preconditioner once we have the approximate eigenvectors. Thus, we proposed an approach, based on relative eigenvector residual norm, to choose well approximated eigenvectors at the end of a restart cycle. This method reduces the number of iterations by a factor up to 7 on our test matrices. We introduced two strategies to use EGMRES as a CPR solver. This solver is used to solve coupled linear systems of equations such as systems arising from simulations of reservoirs. Unlike existing methods, such as proposed in [63], where an algebraic multigrid solves the first level and FGMRES solves the second level, EGMRES is used for the two levels of the coupled system and benefits from the deflation of eigenvalues. A gain in the number of iterations of a factor up to 2 is obtained by CPR-EGMRES compared to CPR-GMRES that uses a direct linear solver, LU, to solve the pressure level. In conclusion, EGMRES reduces the number of iterations to reach convergence. The gain is more important for ill-conditioned linear systems. We compared different thresholds for the criteria of eigenvectors approximation. We noticed that a threshold $\varepsilon = 10^{-2}$ leads to good behavior in general. As future work, the method will be implemented in parallel. We will also test large linear systems on massively parallel computers and compare the method against multigrid and multilevel domain decomposition methods.

CHAPTER 3

DEFLATION SUBSPACES

Outline of the current chapter

This chapter presents deflation strategies related to recycling Krylov subspace methods for solving one or a sequence of linear systems of equations. Besides well-known strategies of deflation, Ritz- and harmonic Ritz-based deflation, we introduce an SVD-based deflation technique. We consider the recycling in two contexts, recalling the Krylov subspace between the cycles of restarts and recycling a deflation subspace when the matrix changes in a sequence of linear systems. Numerical experiments on real-life reservoir simulations demonstrate the impact of our proposed strategy.

3.1 Introduction

We consider the system

$$Ax = b \tag{3.1}$$

where $A \in \mathbb{C}^{n\times n}$ is non-singular and diagonalizable, $b \in \mathbb{C}^n$ is a right-hand side vector and $x \in \mathbb{C}^n$ is the vector of unknowns. We are interested in solving (3.1) by using a Krylov iterative method. GMRES [61], CG [35] and BiCG [56] are widely used Krylov subspace methods. Several works have studied the impact of the spectrum of the matrix A on the convergence rate of Krylov methods [56, 60, 48, 47, 76]. It was noticed that the small and nearly isolated eigenvalues induce a slow rate of convergence. In order to have robustness and fast convergence, a deflation subspace correction can annihilate the impact of this part of the spectrum. Different types of deflation techniques depending on the Hermitian structure of the matrix A are described in the literature [48, 20, 47]. These strategies construct a deflation subspace during the iterative solver. Usually deflation subspaces in recycling methods are approximations of eigenspaces. In [20] the authors propose an approximation by using the Ritz vectors. To preserve the structure of the Arnoldi procedure, Morgan presented the deflation of harmonic Ritz vectors [48]. Few papers discuss the impact of singular vectors on the convergence of Krylov methods. In [65] Simoncini shows the impact of very small singular values on the convergence of several restarted variants of Krylov methods when the right-hand side has a large coefficient in the direction of the left singular vector. This work inspired us to think about deflation of approximate singular vectors rather than eigenvectors.

In several applications, we have to solve successive linear systems $A_i x = b_i$ where $A_i \approx A_{i+1}$, e.g., large eigenvalue problems or the Newton's method for solving nonlinear systems. Furthermore, the Newton's method combined with a Constrained Pressure Residual (CPR [80, 63]) solver have both cases, $A_i \approx A_{i+1}$ and $A_i = A_{i+1}$. Thus, the deflation subspace that is built during the solution of $A_i x = b_i$ can be useful to solve $A_{i+1}x = b_{i+1}$. In [57] the authors present the method GCRO-DR that benefits from previous solutions when solving a sequence of linear systems. In the Hermetian case, Vuik et al. [76] prove that for any deflation subspace the effective condition number of the preconditioned and deflated matrix is smaller than the effective condition number of the preconditioned matrix without deflation. However, this is not true for the non-symmetric case. Moreover, we will see in the numerical experiments 3.6 how the deflation subspace, in several cases might deteriorate the convergence of the method. Thus, when considering a new matrix in the sequence of linear systems, it is important to check whether the deflation subspace is appropriate or not. For this aim, we introduce different strategies to reduce the deflation subspace based on the approximation type of it.

We introduce the deflation based on singular vectors. We generalize the formulation of exact deflation for a symmetric linear system given in [76] to general linear systems. Then we derive the approximation strategy for deflating the singular vectors. We compare the combination of the reduction of the deflation subspace and the deflation of singular vectors to the existing method $GCRO-DR$ on a sequence of linear systems

arising from reservoirs simulations. Since we are interested in solving a sequence of linear systems, we follow the presentation of Parks et al. in [57].

The chapter is organized as follows. In the following section we give a brief review about the Krylov subspace and the Arnoldi procedure, and then we present the deflation subspace correction. Afterwards, we review the method GCRO-DR [57] with an abstract deflation subspace related to the previous search subspace. In Section 3.3 we introduce the deflation of singular vectors. We generalize the formulation of deflation of exact eigenspaces in the Hermitian case [76] to the non-symmetric case. Section 3.4 presents strategies of recovering deflation vectors for GCRO-DR. At first, we recall the Ritz pairs and the harmonic Ritz pairs, and then we introduce a strategy of deflating approximate singular vectors. In Section 3.5 we present the reduction of the deflation subspace in recycling Krylov subspace methods. For each type of deflated vectors we propose a reduction strategy. In Section 3.6 we compare the three variants of deflation techniques combined with reduction of deflation subspace on a sequence of linear systems arising from reservoirs simulations.

3.2 Background

In this section we review the Krylov subspaces, the Arnoldi procedure, and the exact deflation of eigenvectors. We review the GCRO-DR algorithm to which we propose a modification in the following sections.

Notation

For an integer $m > 0$ we refer to the $m \times m$ identity matrix as I_m. Let the integers $m > 0$, $p > 0$ and let $h_{ij} \in \mathbb{C}$ for $i = 1,\ldots,m$, $j = 1,\ldots,p$, we refer to the matrix whose (i,j) coefficient is $h_{i,j}$ for $i = 1,\ldots,m$, $j = 1,\ldots,p$ as $(h_{ij})_{\substack{1\leq i\leq m\\ 1\leq j\leq p}}$. If for some pair (i,j) such that $1 \leq i \leq m$ and $1 \leq j \leq p$ the element $h_{i,j}$ is not defined, it is set to zero. Let $H = (h_{i,j})_{\substack{1\leq i\leq m\\ 1\leq j\leq p}} \in \mathbb{C}^{m\times p}$, we refer to the matrix $(h_{i,j})_{\substack{1\leq i\leq m-1\\ 1\leq j\leq p}} \in \mathbb{C}^{(m-1)\times p}$ as $\overline{H}$. The matrix $(h_{i,j})_{\substack{2\leq i\leq m\\ 1\leq j\leq p}} \in \mathbb{C}^{(m-1)\times p}$ as $\underline{\mathrm{H}}$.

3.2.1 Krylov subspaces and Arnoldi procedure

The Krylov subspace of dimension $k > 0$ associated to A and b is defined as

$$K_k(A,b) = \operatorname{span}\left\{b, Ab, \ldots, A^{k-1}b\right\}. \tag{3.2}$$

Let

$$k_{A,b} = \min_{\substack{P\in\mathcal{P}\\ P(0)=1}} \{deg(P) \mid P(A)b = 0\},$$

where $\mathcal{P}$ represents the set of polynomials with complex coefficients. The vectors $b, Ab, \ldots, A^{k-1}b$ for $k =< k_{A,b}$ form a basis of the Krylov subspace. Due to round-off errors,

an orthogonalization procedure is necessary to construct a basis vectors for the Krylov subspace. To this aim the Arnoldi procedure is widely used, see Algorithm 16.

Algorithm 16 Arnoldi (A, V_1, m)

Require: normal vector $v_1 \in \mathbb{C}^n$, matrix $A \in \mathbb{C}^{n\times n}$, number of iterations m.
Ensure: orthonormal basis vectors. V_{m+1}, Hessenberg matrix $H_m \in \mathbb{C}^{(m+1)\times m}$.

for $j = 1 : m$ **do**
  $w = Av_j$
  for $i = 1 : j$ **do**
    $h_{ij} = v_i^H w$
  end for
  $w = w - \sum_{i=1}^{j} h_{ij} v_i$
  $h_{j+1,j} = \|w\|_2$
  $v_{j+1} = \frac{w}{h_{j+1,j}}$
  $V_m = [v_1, \cdots, v_m]$, $V_{m+1} = [V_m, v_{m+1}]$, $H_m = (h_{ij})_{i,j}$
end for

According to the notations in Algorithm 16 the following relation holds at each iteration k

$$AV_k = V_{k+1}H_k, \tag{3.3}$$

where $V_k = \{v_1, \ldots, v_k\}$, $V_{k+1} = [V_k, v_{k+1}]$, and $H_k = (h_{i,j})_{\substack{1\le i\le k+1\\ 1\le j\le k}}$.

3.2.2 Deflation of eigenvectors

Let S be an A-invariant subspace of dimension $k \geq 0$ related to the smallest eigenvalues of the matrix A. Let $Z \in \mathbb{C}^{n\times k}$ be a matrix whose vectors form an orthonormal basis of the subspace S. Let T be the projection of A on the subspace S, i.e., $T = Z^H AZ$. We note that the following relation holds

$$AZ = ZT.$$

Let $\{\lambda_1, \ldots, \lambda_n\}$ be the eigenvalues of A ordered increasingly by the magnitude of their absolute value, i.e., $|\lambda_1| \leq \ldots \leq |\lambda_n|$. Consider the preconditioned system

$$(I + Z(T^{-1} - I)Z^H)Ax = (I + Z(T^{-1} - I)Z^H)b \tag{3.4}$$

The following Proposition 3 can be found in [20]. It describes how to deflate the eigenvalues of the matrix A by adding a low-rank correction.

Theorem 3. *The eigenvalues of the matrix* $A + Z(T^{-1} - I)Z^H A$ *are* $\{1, \ldots, 1, \lambda_{k+1}, \ldots, \lambda_n\}$.

The equation (3.4) can be rewritten as:

$$\left(\left(A - ZZ^H A\right) + ZT^{-1}Z^H A\right)x = \left(\left(b - ZZ^H b\right) + ZT^{-1}Z^H b\right)$$

In other words, to solve (3.1), we look for the solution in two subspaces, the first is inside S, the second is outside S. The first finds the part of the solution that is generated by the basis vectors V. This is done by solving the deflation subspace problem with the matrix T. The second finds the part that is orthogonal to S. This part is left to the Krylov method. Since the eigenvectors of the matrix are not orthogonal in general, we need at each iteration of the Krylov method to correct the new computed basis vector. In [76], the authors present how to avoid doing this correction at each iteration when the considered matrix is Hermitian Positive Definite (HPD).

3.2.3 Ritz pairs

In this paragraph we review the definition of Ritz pairs of a matrix related to a subspace.

Definition 1. *Let $B \in \mathbb{C}^{n\times n}$ and let $\mathcal{S}$ be a subspace of dimension $k < n$. We say that the pair $(u,\theta) \in \mathbb{C}^n \times \mathbb{C}$ is a Ritz pair of B related to the subspace $\mathcal{S}$ if*

$$\begin{cases} u \in \mathcal{S}, \\ y^H(Bu - \theta u) = 0, \forall y \in S. \end{cases} \tag{3.5}$$

In the case where the subspace $\mathcal{S} = K_k(A,b)$, the Ritz pairs of the matrix A correspond to

$$\mathcal{R} = \{(V_k u, \theta), \overline{H}_k u = \theta u\},$$

see Algorithm 16. If we consider the subspace $\mathcal{S} = AK_k(A,b)$, the Ritz pairs of the matrix A^{-1} correspond to

$$\mathcal{R} = \{(V_k u, \theta), \overline{H}_k^H u = \theta H_k^H H_k u\}.$$

Lemma 3. *Let $B \in \mathbb{C}^{m\times m}$ and let S be a subspace of dimension $k < m$ that contains an eigenpair (λ, u) of the matrix B. Let $V \in \mathbb{C}^{m\times k}$ be a matrix whose columns form an orthonormal basis of S. Then, there exists a vector $w \in \mathbb{C}^k$ such that (λ, w) is an eigenpair of $V^H BV$, the restriction of B to the subspace S. Furthermore, $w = V^H u$.*

Proof. The proof is simple and direct. □

In Section 3.4 we will review in detail the derivation of these Ritz pairs. Furthermore, we derive the Ritz pairs relative to the singular vectors and values of A.

In the following we review the GCRO-DR method and the construction of an abstract deflation subspace related to it.

3.2.4 GCRO-DR

We recall that CGRO-DR is an iterative method for solving a sequence of linear systems which takes advantage of the history of previous search subspaces by performing recycling. In [57], the authors present the algorithm with a deflation subspace related to the harmonic Ritz vectors. We explain briefly the method. Given the sequence of linear systems $A_i x = b_i$, for $i = 1,\dots,p$, where $p > 1$, the method GCRO-DR(m,k) works

as the following. To solve the linear system $A_1 x = b_1$, the first cycle is performed as a usual GMRES cycle. The subspace relation that holds at the end of the cycle is the usual Arnoldi relation:

$$A_1 V_m = V_{m+1} H_m.$$

The solution x_1 and the residual r_1 are computed as usual. Based on a given strategy, a column matrix $P_k \in \mathbb{C}^{m \times k}$ is computed (Section 3.4 details the computation of this matrix following different strategies). This matrix is expanded by V_m to form the basis of the deflation subspace $\tilde{Y}_k = V_m P_k$. The image of the deflation subspace by A_1 is computed

$$A_1 V_m P_k = V_{m+1} H_m P_k,$$

then orthonormalized,

$$C_k = V_{m+1} Q,$$

where Q stands for the unitary factor of the QR factorization of $H_m P_k = QR$. The relation that holds at this stage is:

$$A_1 \tilde{Y}_k R^{-1} = C_k.$$

We note $U_k = \tilde{Y}_k R^{-1}$. In order to avoid round-off errors, the vectors of U_k are normalized. We note D_k the diagonal matrix such that the following holds

$$\begin{aligned} \tilde{U}_k &= U_k D_k^{-1}, \\ A_1 \tilde{U}_k &= C_k D_k. \end{aligned}$$

At that moment, the method is ready to continue to the next cycle, $m-k$ Arnoldi iterations are performed with $(I - CC^H)A$ as an operator and the normalized residual as the starting vector. The following relation holds

$$A_1 [\tilde{U}_k, V_{m-k}] = [C_k, V_{m-k+1}] G_m,$$

where V_{m-k+1} is the resulting orthonormal basis constructed by the Arnoldi procedure, $G_m = \begin{pmatrix} D_k & B_{m-k} \\ 0 & H_{m-k} \end{pmatrix}$, $B_{m-k} = C_k^H A_1 V_{m-k}$ and H_{m-k} is the resulting Hessenberg matrix of the Arnoldi procedure. The solution x and the residual r are computed following the constraint $\|r_2\|_2 = \|r_1 - A_1 x\|_2$ has a minimal norm over the subspace spanned by the columns of $[\tilde{U}_k, V_{m-k}]$. The extraction of the deflation subspace is performed as before by replacing the matrix H_m by G_m. This cycle is repeated until the convergence is achieved. In order to solve the second linear system, all what is necessary is to update the image of the deflation subspace, i.e., C_k, by the image of the deflation subspace by the new matrix A_2. Algorithm 17 presents the GCRO-DR method with an abstract deflation subspace. We will discuss the deflation variants in Section 3.4.

Considering the notations in Algorithm 17, if the matrix $\tilde{Y}_k$ is not defined, then (3.3)

Algorithm 17 GCRO-DR

Require: the maximum dimension of the search subspace m, the dimension of the deflation subspace k, the convergence tolerance ε, the initial guess x_0.
Ensure: approximate solution $\tilde{x}$ of $Ax = b$.

initialization $r_0 = b - Ax_0$, $i = 1$
if $\tilde{Y}_k$ is defined (from solving a previous linear system) **then**
  let $[Q, R]$ be the QR factorization of $A\tilde{Y}_k$
  $C_k = Q$
  $U_k = \tilde{Y}_k R^{-1}$
  $x_1 = x_0 + U_k C_k^H r_0$
  $r_1 = r_0 - C_k C_k^H r_0$
else
  $v_1 = r_1 / \| r_1 \|$
  $c = \|r_0\|_2 e_1$
  perform m steps of GMRES, solving $\min \|c - H_m y\|_2$ for y and generating V_{m+1} and H_m
  $x_1 = x_0 + V_m y$
  $r_1 = V_{m+1}(c - H_m y)$
  compute k deflation vectors P_k
  $\tilde{Y}_k = V_m P_k$
  let $[Q, R]$ be the QR factorization of $H_m P_k$
  $C_k = V_{m+1} Q$
  $U_k = \tilde{Y}_k R^{-1}$
end if
while $\|r_i\|_2 > \varepsilon$ **do**
  $i = i + 1$
  perform $m - k$ Arnoldi steps with the linear operator $(I - C_k C_k^H)A$, letting $v_1 = r_{i-1} / \|r_{i-1}\|_2$ and generating V_{m-k+1}, H_{m-k}, and $B_{m-k} = C_k^H A V_{m-k}$
  let D_k be a diagonal scaling matrix such that $\tilde{U}_k = U_k D_k$, where the columns of $\tilde{U}_k$ have a unit norm
  $\hat{V}_m = [\tilde{U}_k, V_{m-k}]$, $\hat{W}_{m+1} = [C_k, V_{m-k+1}]$
  $G_m = \begin{pmatrix} D_k & B_{m-k} \\ 0 & H_{m-k} \end{pmatrix}$
  solve $min \| G_m y - \hat{W}_{m+1}^H r_{i-1} \|_2$ for y
  $x_i = x_{i-1} + V_m y$
  $r_i = r_{i-1} - \hat{W}_{m+1} G_m y$
  compute k deflation vectors P_k
  $\tilde{Y}_k = \hat{V}_m P_k$
  let $[Q, R]$ be the QR factorization of $G_m P_k$
  $C_k = \hat{W}_{m+1} Q$
  $U_k = \tilde{Y}_k R^{-1}$
end while
let $\tilde{Y}_k = U_k$

holds. Otherwise, the following relation holds

$$A[\tilde{U}_k, V_{m-k}] = [C_k, V_{m-k+1}]\begin{pmatrix} D_k & B_{m-k} \\ 0 & H_{m-k} \end{pmatrix}. \tag{3.6}$$

The matrix $[C_k, V_{m-k+1}]$ is unitary, but this is not true, in general, for the matrix $[\tilde{U}_k, V_{m-k}]$. In this case the subspace generated by the columns of $\tilde{U}_k, V_{m-k}$ is not a Krylov subspace.

In the following section we discuss the deflation of exact singular vectors.

3.3 Deflation based on singular vectors

As previously mentioned, the work of Simoncini on the convergence of several restarted Krylov methods and its relation with the singular vectors associated to the smallest singular values [65] motivated us to investigate the deflation based on singular vectors approximation. This section introduces the deflation of singular vectors. Given a set of singular vectors (right or left), we show how to deflate these vectors during the Krylov method.

Theorem 4. *Let x_* be the exact solution of (3.1). Let*

$$A = U\Sigma V^H$$

be the singular value decomposition of A such that the singular values are ordered increasingly. Let k_τ be the number of singular values smaller than a given threshold $\tau > 0$. Consider the splitting of the SVD decomposition

$$A = U_1\Sigma_1 V_1^H + U_2\Sigma_2 V_2^H,$$

where $\Sigma_2 \in \mathbb{C}^{k_\tau \times k_\tau}$ is a diagonal matrix whose diagonal elements are the singular values smaller than τ. Consider $\tilde{x}$ an approximate solution of the following linear system of equations

$$(I - U_2U_2^H)Ax = (I - U_2U_2^H)b, \tag{3.7}$$

such that $\|\hat{x} - \tilde{x}\|_2 \leq \varepsilon$, where $\hat{x}$ is an exact solution of (3.7) and $\varepsilon > 0$. Then, the following holds

$$\|x_* - (I - V_2V_2^H)\tilde{x} - V_2\Sigma_2^{-1}U_2^Hb\|_2 \leq \varepsilon.$$

Proof. First, we remark that x_* is a solution of (3.7) and the set of solutions of (3.7) can be written as $\mathcal{S} = \{x = x_* + V_2u,\ u \in \mathbb{C}^{k_\tau}\}$. Indeed, let x be a solution of (3.7). Then,

$$Ax = (I - U_2U_2^H)b + U_2U_2^HAx,$$
$$Ax = Ax_* - U_2U_2^H(b - Ax).$$

We multiply by A^{-1} both sides. We obtain

$$\begin{aligned} x &= x_* + V_2\Sigma_2^{-1}U_2^H(Ax-b) \\ &= x_* + V_2\Sigma_2^{-1}(\Sigma_2 V_2^H x - U_2^H b) \\ &= x_* + V_2V_2^H x - V_2\Sigma_2^{-1}U_2^H b. \end{aligned}$$

On the other side, each element of $\mathcal{S}$ is a solution of (3.7).

We have

$$\begin{aligned} V_2V_2^H x_* &= V_2V_2^H A^{-1}b \\ &= V_2\Sigma_2^{-1}U_2^H b. \end{aligned}$$

Hence, we can write

$$\begin{aligned} x_* &= (I - V_2V_2^H)x_* + V_2V_2^H x_*, \\ &= (I - V_2V_2^H)\hat{x} + V_2\Sigma_2^{-1}U_2^H b, \end{aligned}$$

thus,

$$x_* - (I - V_2V_2^H)\tilde{x} - V_2\Sigma_2^{-1}U_2^H b = (I - V_2V_2^H)(\hat{x} - \tilde{x}).$$

Finally, we obtain

$$\begin{aligned} \|x_* - (I - V_2V_2^H)\tilde{x} - V_2\Sigma_2^{-1}U_2^H b\|_2 &\leq \|(I - V_2V_2^H)(\hat{x} - \tilde{x})\|_2, \\ &\leq \|\hat{x} - \tilde{x}\|_2, \\ &\leq \varepsilon. \end{aligned}$$

□

Note that the matrix $(I - U_2U_2^H)A$ is singular. However, solving the linear system of equations (3.7) with a Krylov method is possible. The approximate solution of (3.1) can be recovered as

$$x = (I - V_2V_2^H)\tilde{x} + V_2x_2,$$

where $x_2 = \Sigma_2^{-1}U_2^H b$ and $\tilde{x}$ is the approximate solution given by the Krylov method.

The previous discussion is a generalization of the theory of deflation of eigenvectors for an SPD matrix to a general invertible matrix. In the next section we discuss different approaches for recovering deflation information from a Krylov subspace. Two types are based on approximating eigenvectors of the matrix, that can be found in literature [48, 57]. A third type is introduced here and it is based on approximating the right singular vectors of the matrix.

3.4 Recovering deflation vectors

In this section we derive three different types of deflation subspaces. These subspaces are considered in the context of GCRO-DR [57]. In Section 3.2 we reviewed the method GCRO-DR for an abstract deflation subspace issued from the last Krylov subspace. Even though any deflation vectors can be chosen for GCRO-DR, this choice has an impact on the convergence of the method. Two types of deflation vectors have been presented in previous works [48, 20, 57]. They are based on the smallest and largest Ritz values of the matrices A, A^{-1} respectively. During the GCRO-DR method one of the relations (3.3) and (3.6) holds. In the following we present how to recover the deflation vectors mentioned above. These strategies are introduced previously in previous works [48, 20, 57]. We present them for the sake of completeness. Afterwards, we introduce the deflation vectors that are related to the right singular vectors.

We consider the following environment. Let K be a search subspace (Krylov subspace or else) of dimension m associated to the matrix A, let $V \in \mathbb{C}^{n\times m}$ be a matrix whose columns form a basis of K. Let $W \in \mathbb{C}^{n\times(m+1)}$ be a unitary matrix such that the following relation holds

$$AV = WG, \tag{3.8}$$

where $G \in \mathbb{C}^{(m+1)\times m}$ is a Hessenberg matrix.

We will derive the eigenvalue problem that is solved in order to compute the deflation vectors. For each approximation strategy, we formulate the computation following the notations in Algorithm 17. Thus, we will distinguish two cases in each strategy, depending on whether the deflation subspace already exists or not.

3.4.1 Approximation based on the smallest Ritz values of A

In this case the Ritz pairs (u, θ) verify

$$\begin{cases} u \in K, \\ y^H(Au - \theta u) = 0, \forall y \in K. \end{cases}$$

Since the columns of V span the subspace K, we have

$$V^\top(AVw - \theta Vw) = 0,$$

where $u = Vw$. This can be written as

$$V^\top WGw = \theta V^\top Vw.$$

By adapting the notations in Algorithm 17, the deflation vectors can be recovered by solving the following eigenvalue problem

$$\begin{cases} \overline{H}_m w = \theta w, & \text{if } \tilde{Y}_k \text{ is not defined} \\ \begin{pmatrix} \tilde{U}_k^H C_k & \tilde{U}_k^H V_{m-k+1} \\ 0 & (I_{m-k}, 0) \end{pmatrix} G_m w = \theta \begin{pmatrix} \tilde{U}_k^H \tilde{U}_k & \tilde{U}_k^H V_{m-k} \\ V_{m-k}^H \tilde{U}_k & I_{m-k} \end{pmatrix} w, & \text{if } \tilde{Y}_k \text{ is defined.} \end{cases} \tag{3.9}$$

3.4.2 Approximation based on the largest Ritz values of A^{-1}

To approximate the eigenvectors of A^{-1} we look for the approximate vectors in the subspace AK. We write

$$\begin{cases} u \in AK, \\ y \perp (A^{-1}u - \theta u), \forall y \in AK, \end{cases}$$

The columns of AV span the subspace AK. Hence, we have

$$V^H A^H (Vw - \theta AVw) = 0,$$

this can be written as

$$G^H W^H V w = \theta G^H G w.$$

By using the notations of Algorithm 17, this requires solving the following generalized eigenvalue problem

$$\begin{cases} \overline{H}_m^H w = \theta H_m^H H_m w, & \text{if } \tilde{Y}_k \text{ is not defined} \\ G_m^H \begin{pmatrix} C_k^H \tilde{U}_k & 0 \\ V_{m-k+1} \tilde{U}_k & \begin{pmatrix} I_{m-k} \\ 0 \end{pmatrix} \end{pmatrix} w = \theta G_m^H G_m w, & \text{if } \tilde{Y}_k \text{ is defined.} \end{cases} \tag{3.10}$$

We note that Morgan, in his paper [48], refers to the values θ^{-1} as the *harmonic* Ritz values.

3.4.3 Approximation based on the smallest Ritz values of $A^H A$

We follow the definition of the Ritz pairs Definition 1. In order to approximate the left singular vectors of the matrix A, we consider an approximation of the eigenvectors of the matrix $A^H A$. Thus, we are looking for the pair (u, θ) such that

$$\begin{aligned} &u \in K, \\ &y \perp (A^H A u - \theta u), \forall y \in K \end{aligned}$$

The columns of V span the subspace K, the previous relation can be written as

$$V^H(A^H A V w - \theta V w) = 0,$$

where $u = Vw$. This can be written as

$$G^H G w = \theta V^H V w.$$

The formulation of this eigenvalue problem in the context of Algorithm 17 is as follows

$$\begin{cases} H_m^H H_m w = \theta w, & \text{if } \tilde{Y}_k \text{ is not defined} \\ G_m^H G_m w = \theta \begin{pmatrix} \tilde{U}_k^H \tilde{U}_k & 0 \\ 0 & I_{m-k} \end{pmatrix} w, & \text{if } \tilde{Y}_k \text{ is defined.} \end{cases} \tag{3.11}$$

3.5 Deflation subspace reduction

In this section we introduce a strategy to reduce the dimension of the deflation subspace. In a sequence of linear systems solved by GCRO-DR and after solving a linear system, the deflation subspace is used to solve the following linear system. Since the matrices can be different with no information about the change, the deflation subspace may not be useful anymore. In the Hermitian case, the deflation subspace is at least useless. However, this is not the case for the non-Hermitian matrices. The deflation subspace might deteriorate the convergence. Thus, we need a strategy to reduce the dimension of the deflation subspace judiciously if necessary. This strategy should respect an important constraint. A good deflation vector should be kept in the reduced deflation subspace. Good, here, means that the vector approximates well an eigenvector of the matrix that defines the deflation strategy and the associated eigenvalue is small in the case of (A, $A^H A$) and large in the case of A^{-1}. The motivation to this strategy is the following. Let B be the matrix related to the deflation strategy (e.g., A). Suppose that the deflation subspace contains an exact eigenvector u of B associated to the eigenvalue $\theta < \tau$, where $\tau > 0$ is a predefined value. Lemma 3 proves that the vector u has a corresponding eigenvector of the projected matrix to the deflation subspace. The threshold τ can be chosen to be relative either to the largest approximated eigenvalue or the largest approximated singular value following the strategy of deflation.

Hence, for each strategy of recycling, presented in Section 3.4, the reduction strategy is associated to it, i.e., if the recycling strategy deflates the Ritz vectors associated to the small (resp. large) Ritz values (corresponding to the search subspace), the reduction strategy disregards the Ritz vectors associated to the large (resp. small) Ritz values (corresponding to the deflation subspace). The reduced subspace is a subset of the original deflation subspace.

In the following we derive the generalized eigenvalue problems that are necessary to perform the reduction of the deflation subspace. We suppose that the columns of the unitary matrix Z span the deflation subspace $\mathcal{S}$. Table 3.1 presents the eigenvalue

problems to solve in order to reduce the deflation subspace corresponding to each strategy of deflation.

Reference matrix	Eigenvalue problem	Disregarded part
A	$Z^H AZw = \theta w$	largest magnitude
A^{-1}	$Z^H A^H Zw = \theta Z^H A^H AZw$	smallest magnitude
$A^H A$	$Z^H A^H AZw = \theta w$	largest magnitude

Table 3.1: Generalized eigenvalue problems to reduce the deflation subspace, *Reference matrix* stands for the matrix whose eigenvectors are approximated, *Eigenvalue problem* refers to the problem to solve in order to reduce the deflation subspace, *Disregarded part* refers to the part of the spectrum to which are associated the disregarded vectors.

We note that for the case of an approximation of singular vectors it is more robust to perform an SVD factorization of AZ rather than solving the eigenvalue problem $Z^H A^H AZw = \theta w$.

Algorithm 18 Deflation subspace reduction

Require: matrix A, deflation basis vectors $\tilde{Y}$, threshold τ, def the matrix to which the deflation subspace is related
Ensure: reduced deflation basis vectors $\tilde{Y}$
1: compute the QR factorization of $\tilde{Y} = ZR$
2: **if** $def = A$ **then**
3: solve $Z^H AZw = \theta w$
4: **else if** $def = A^{-1}$ **then**
5: solve $Z^H A^H AZw = \theta Z^H A^H Zw$
6: **else if** $def = A^H A$ **then**
7: solve $Z^H A^H AZw = \theta^2 w$
8: **end if**
9: form $\tilde{Y}$ whose columns are Zw where w is an eigenvector associated $\theta < \tau$

Here we propose the method Generalized Minimal Residual with Modified Deflated Restarting (GMRES-MDR), Algorithm 19. This method is based on the GCRO-DR method. The main difference is that an estimation of the previous deflation subspace (when the matrix in the sequence changes) is performed in order to keep only necessary vectors.

3.6 Numerical experiments

In this section we illustrate the impact of the proposed method on the convergence of sequences of linear systems arising from reservoirs simulations. We present sequential experiments performed in MATLAB. Our set of test cases are obtained from the in-house prototype code at Total, which simulates a complex enhanced oil recovery (EOR)

Algorithm 19 GMRES-MDR

Require: the maximum dimension of the search subspace m, the maximum dimension of the deflation subspace k_{max}, the convergence tolerance ε, the initial guess x_0, the matrix to which the deflation subspace is related def, τ threshold of reduction of the deflation subspace.

Ensure: approximate solution $\tilde{x}$ of $Ax = b$.

initialization $r_0 = b - Ax_0$, $i = 1$, $k = k_{max}$
if $\tilde{Y}_k$ is defined (from solving a previous linear system) **then**
  call Algorithms 18 and get $\tilde{Y}_k$, where k is the number of columns in $\tilde{Y}_k$
  let $[Q, R]$ be the QR factorization of $A\tilde{Y}_k$
  $C_k = Q$
  $U_k = \tilde{Y}_k R^{-1}$
  $x_1 = x_0 + U_k C_k^H r_0$
  $r_1 = r_0 - C_k C_k^H r_0$
else
  $v_1 = r_1 / \| r_1 \|$
  $c = \|r_0\|_2 e_1$
  perform m steps of GMRES, solving $\min \|c - H_m y\|_2$ for y and generating V_{m+1} and H_m
  $x_1 = x_0 + V_m y$
  $r_1 = V_{m+1}(c - H_m y)$
  compute k deflation vectors P_k following def and Section 3.4
  $\tilde{Y}_k = V_m P_k$
  let $[Q, R]$ be the QR factorization of $H_m P_k$
  $C_k = V_{m+1} Q$
  $U_k = \tilde{Y}_k R^{-1}$
end if
while $\|r_i\|_2 > \varepsilon$ **do**
  $i = i + 1$
  perform $m - k$ Arnoldi steps with the linear operator $(I - C_k C_k^H)A$, letting $v_1 = r_{i-1} / \|r_{i-1}\|_2$ and generating V_{m-k+1}, H_{m-k}, and $B_{m-k} = C_k^H A V_{m-k}$
  let D_k be a diagonal scaling matrix such that $\tilde{U}_k = U_k D_k$, where the columns of $\tilde{U}_k$ have a unit norm
  $\hat{V}_m = [\tilde{U}_k, V_{m-k}]$, $\hat{W}_{m+1} = [C_k, V_{m-k+1}]$
  $G_m = \begin{pmatrix} D_k & B_{m-k} \\ 0 & H_{m-k} \end{pmatrix}$
  solve $min \| G_m y - \hat{W}_{m+1}^H r_{i-1} \|_2$ for y
  $x_i = x_{i-1} + V_m y$
  $r_i = r_{i-1} - \hat{W}_{m+1} G_m y$
  set $k = k_{max}$ compute k deflation vectors P_k following def and Section 3.4
  $\tilde{Y}_k = \hat{V}_m P_k$
  let $[Q, R]$ be the QR factorization of $G_m P_k$
  $C_k = \hat{W}_{m+1} Q$
  $U_k = \tilde{Y}_k R^{-1}$
end while
let $\tilde{Y}_k = U_k$

mechanism. This simulator relies on a finite volume discretization and a two-point flux approximation. We have two sequences of linear systems denoted with initials BIGCO24 and BIGP1. BIGCO24 corresponds to a simulation of water and gaz injection using a compositional model (8 hydrocarbon components). The permeability field is heterogeneous. The grid has 83587 active cells. BIGP1 comes from the simulation of water injection using a black-oil model. The permeability field is heterogeneous (sector model from a real field case). The grid has 42332 active cells.

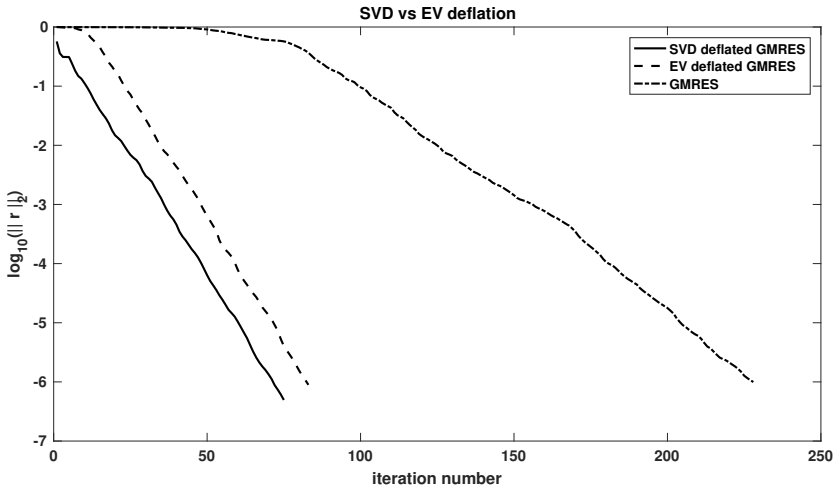

Figure 3.1: Convergence history, singular vectors deflated GMRES against eigenvectors deflated GMRES. The convergence history of non-deflated GMRES is also plotted, results for the matrix BIGP1 on MATLAB 2017. The 20 computed approximated vectors (singular vectors and eigenvectors) are approximated up to a tolerance of 10^{-6}

Figure 3.1 shows the impact of deflating the left singular vectors related to the smallest singular values. In each variant 20 vectors are computed approximately to a tolerance of 10^{-6}. The deflation of the eigenvectors corresponds to (3.4) that we rewrite here,

$$(I + Z(T^{-1} - I)Z^H)Ax = (I + Z(T^{-1} - I)Z^H)b,$$

where Z is a unitary matrix whose columns span the apprximated eigenspace. The deflation of the singular vectors follows the Theorem 4. We write the preconditioned linear system

$$(I - U_2U_2^H)Ax = (I - U_2U_2^H)b,$$

where $U_2 = AV_2\Sigma_2^{-1}$ stands for the apprximated left singular vectors. The solution is recovered as $x = (I - V_2V_2^H)\bar{x} + V_2x_2$, where $\bar{x}$ is the solution obtained by GMRES and $x_2 = \Sigma_2^{-1}U_2^Hb$. We remark that the speed of convergence for both deflated variants are approximately the same with a small advantage for the SVD deflation. Furthermore, in the first iterations the SVD deflated GMRES has a sharp convergence curve while

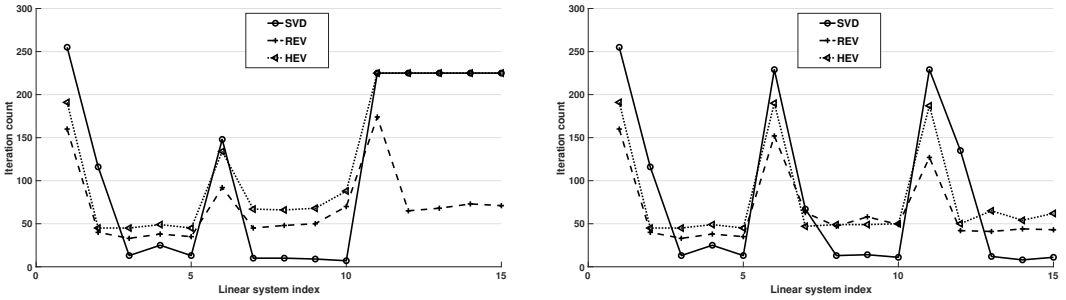

Figure 3.2: Comparison between three variants of deflation subspaces. SVD, REV, and HEV stand for the deflation method related to the matrices $A^H A$, A and A^{-1}, respectively. The maximum dimension of the search subspace is 30. The maximum dimension of the deflation subspace is 5. Fifteen linear systems in the sequence BIGP1, three matrices each one 5 times, with different right-hand sides (for index i the three methods share the same right-hand side). On the left no reduction of the deflation subspace is performed, on the right a reduction of the deflation subspace is performed with a threshold 10^{-3}

the curve corresponding to the eigenvectors deflated GMRES stagnates for a number of iterations. This phenomenon is important for our application. In each Newton iteration during the simulation of a reservoir the linearized system is solved by using the CPR solver. An iteration of the latter solves the linear system associated to the pressure variable. This matrix does not change during the CPR solve, only the right-hand side changes. In practice, a large threshold of convergence for the pressure level is sufficient. Thus, having stagnation at the beginning of the method, even for a small number of iterations, can impact the global number of iterations over the entire simulation. The relative tolerance of convergence in the experiments related to the sequences of linear systems is fixed to 10^{-1}.

As it was noted in the thesis of Parks [58], the choice of the deflation subspace has an impact on the convergence of sequences of linear systems. Figure 3.2 does not only confirm this note but also shows how it might be possible that in some configurations the deflation subspace becomes a problem and leads to the stagnation of the residual norm. We solve thirty linear systems related to three newton steps for each deflation strategy. Each newton step is solved by ten iterations of a CPR solver. In order to make a fair comparison and avoid the eventual large components of the right-hand sides on special eigenvectors or singular vectors, we generate random right-hand sides with seeds corresponding to the index of the linear system, from one to thirty. In the same figure we show the impact of the reduction of the deflation subspace. Figure 3.3 shows how the reduction strategy keeps the efficiency of the method when the existing deflation subspace is useful. There might be some unnecessary small increase in the number of iterations due to reducing the deflation subspace. It is possible to apply the reduction strategy after one restart cycle of a new system. This also helps measure the efficiency of the deflation subspace based on the relative residual norm at the end of this cycle.

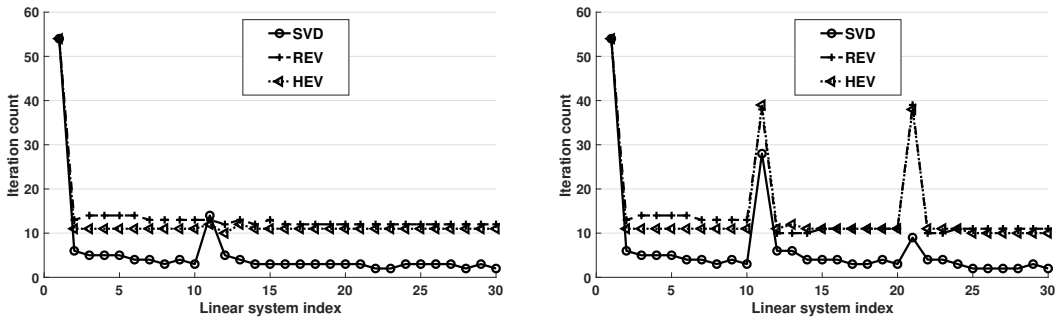

Figure 3.3: Comparison between three variants of deflation subspaces. SVD, REV, and HEV stand for the deflation method related to the matrices $A^H A$, A and A^{-1}, respectively. The maximum dimension of the search subspace is 30. The maximum dimension of the deflation subspace is 5. Thirty linear systems in the sequence BIGCO24, three matrices each one 10 times, with different right-hand sides (for index i the three methods share the same right-hand side). On the left no reduction of the deflation subspace is performed, on the right a reduction of the deflation subspace is performed with a threshold 10^{-3}

Either we keep it or we reduce it.

We notice two things in our numerical experiments. The first is that deflating the Ritz pairs related to A is efficient when no deflation subspace exists. The second is that deflating the Ritz pairs related to $A^H A$ needs more iterations that the one related to A in order to be efficient.

To benefit from both subspaces we propose a simple criterion based on the relative residual during the restart cycle to choose adaptively the subspace of deflation. Figures 3.4 and 3.5 present numerical experiments on the same two previous sequences.

Figures 3.6 and 3.7 present the history of the residual norm corresponding to each strategy of deflation. As we noted previously, we notice that the reduction of the residual norm starting from the first iteration makes the SVD variant has more advantage against other strategies. The total iteration numbers in the BIGCO24 sequence corresponding to the deflation related to A, A^{-1}, and the alternating (A, $A^H A$) is 418, 374, and 163, respectively. We also notice that using the adaptive alternate deflation between the Ritz pairs of A and $A^H A$ has the benefit of both strategies, quick useful deflation subspace and quick reduction of residual norm. Moreover in this example we see how the residual norm stagnates starting from the third matrix systems with the harmonic Ritz-based deflation subspace. The total iteration numbers in the BIGP1 sequence corresponding to the deflation strategies related to A, and the alternating (A, $A^H A$) is 1062 and 686, respectively.

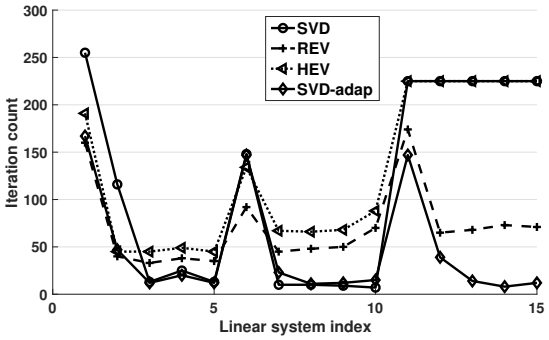

Figure 3.4: Comparison between four variants of deflation subspaces. SVD, REV, and HEV stand for the deflation method related to the matrices $A^H A$, A, and A^{-1}, respectively. SVD-adap stands for an adaptive strategy that alternates the deflation between SVD and REV based on the relative residual norm during the cycle. Fifteen linear systems in the sequence BIGP1, three matrices each one 5 times, with different right-hand sides (for index i the three methods share the same right-hand side).

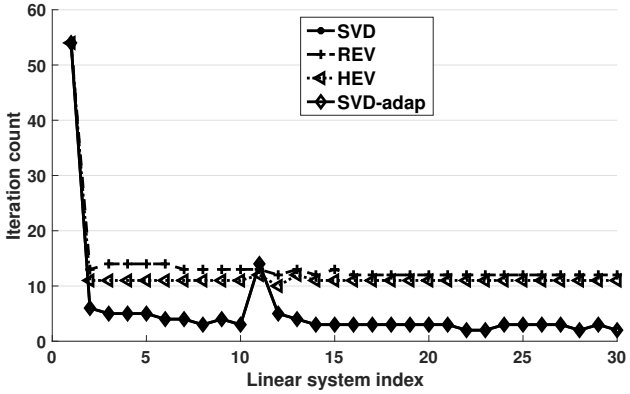

Figure 3.5: Comparison between three variants of deflation subspaces. SVD, REV, and HEV stand for the deflation method related to the matrices $A^H A$, A and A^{-1}, respectively. SVD-adap stands for an adaptive strategy that alternates the deflation between SVD and REV based on the relative residual norm during the cycle. The maximum dimension of the search subspace is 30. The maximum dimension of the deflation subspace is 5. Thirty linear systems in the sequence BIGCO24, three matrices each one 10 times, with different right-hand sides (for index i the three methods share the same right-hand side).

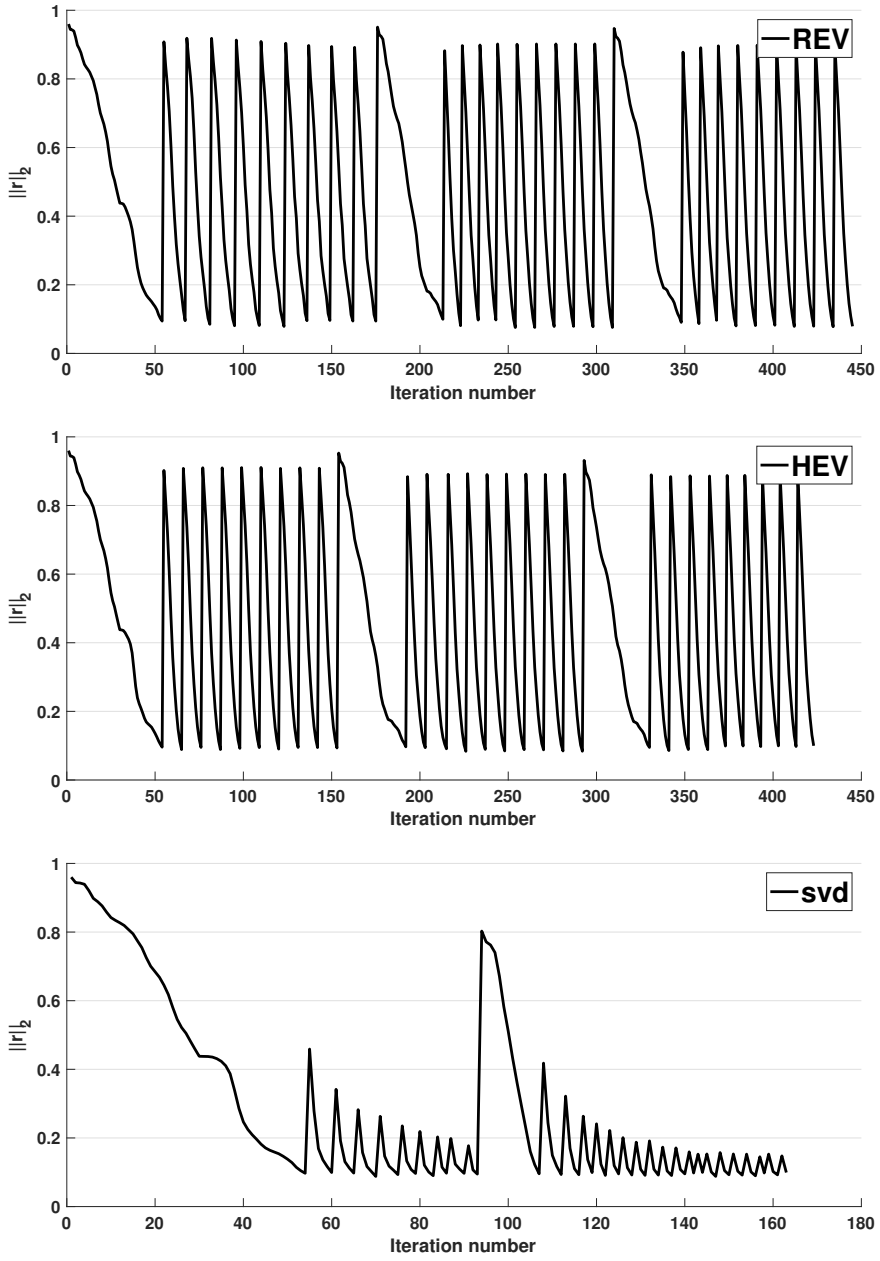

Figure 3.6: History of residual norm in a sequence of linear systems (a peak corresponds to the beginning of a new linear system). Comparison between three variants of deflation subspaces. SVD, REV, and HEV stand for the deflation method related to the matrices $A^H A$, A and A^{-1}, respectively. The SVD variant uses an adaptive strategy that alternates the deflation between SVD and REV deflation methods based on the relative residual norm during the cycle. The maximum dimension of the search subspace is 30. The maximum dimension of the deflation subspace is 5. Thirty linear systems in the sequence BIGCO24, three matrices each one 10 times, with different right-hand sides (for index i the three methods share the same right-hand side).

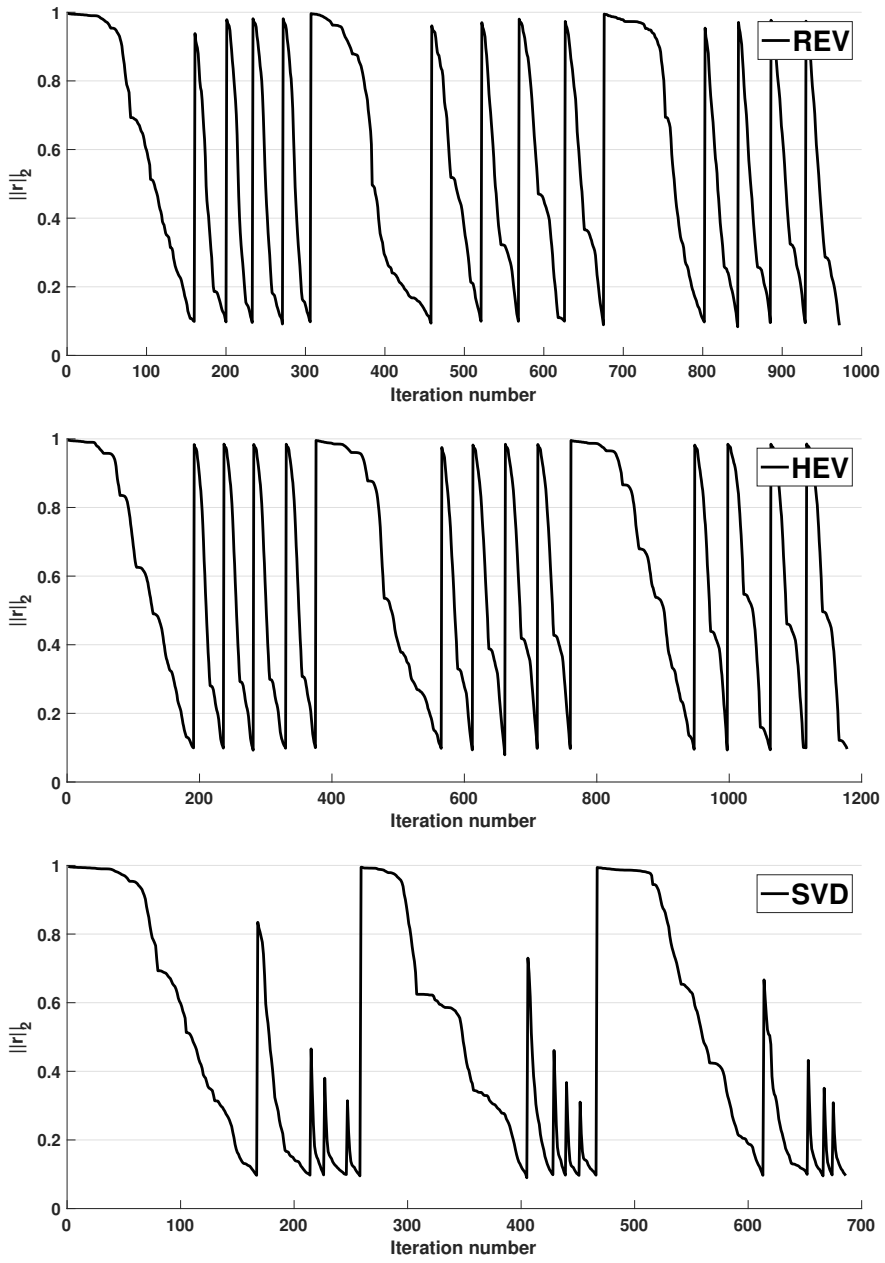

Figure 3.7: History of residual norm in a sequence of linear systems (a peak corresponds to the beginning of a new linear system). Comparison between three variants of deflation subspaces. SVD, REV, and HEV stand for the deflation method related to the matrices $A^H A$, A and A^{-1}, respectively. The SVD variant uses an adaptive strategy that alternates the deflation between SVD and REV based on the relative residual norm during the cycle. The maximum dimension of the search subspace is 30. The maximum dimension of the deflation subspace is 5. Fifteen linear systems in the sequence BIGP1, three matrices each one 5 times, with different right-hand sides (for index i the three methods share the same right-hand side).

3.7 Conclusion

In this chapter we reviewed the deflation concept in Krylov iterative methods. Usual choice of deflation subspaces is related to the approximation of eigenvectors. Based on the study in [65] we introduced the deflation of singular vectors. Since we are interested in solving sequence of linear systems, we reviewed the method GCRO-DR [57] with an abstract deflation subspace. Two previously introduced subspaces were reintroduced. Afterwards, we proposed a deflation subspace related to an approximation of singular vectors. To validate the proposed strategy, a comparison between the three variants was made on different sequences arising from reservoirs simulations. This validation was performed on a sequential code. The numerical experiments demonstrated the impact of choosing the SVD variant. The gain in the global iteration count is up to $\approx 40\%$.

CHAPTER 4

PARALLEL IMPLEMENTATION

Outline of the current chapter

In this chapter we describe the parallel scheme of our implementation of the methods RD-EGMRES and GMRES-MDR. The common part of two implementations is given at the beginning. Then, the details related to each implementation are treated separately.

4.1 Common part

At first we describe the distribution of data over processors. Then, we detail the routines that we use in our implementation.

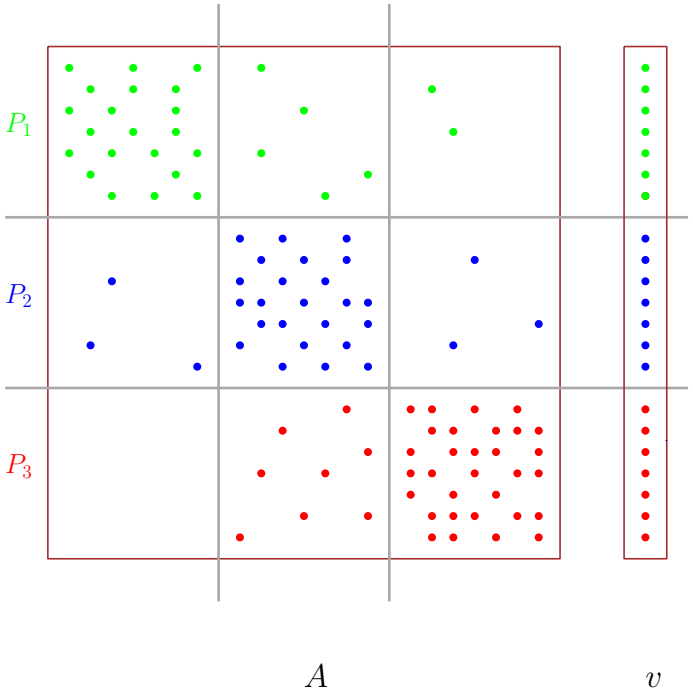

Figure 4.1: Illustration of the distribution of the sparse matrix operator and the vectors over processors

4.1.1 Data distribution

Matrices and vectors in our implementation can be separated into two types related to the locality, distribution and redundancy. The matrix A and the search subspace (Krylov or deflation) vectors are distributed as displayed in Figure 4.1. All matrices and vectors whose both dimensions are less than or equal to the dimension of the search subspace are redundant. Each processor has a local access to this data. Using this distribution, the application of the sparse operator requires communication between neighbour processors. Only dot product operations require global communication.

In the following we describe how data is stored in memory.

Memory management

Dense matrices in iterative solvers are usually stored in one of two ways, either in column major or in row major. We use column major format due to constraints in the reservoir simulator. The vectors of the search subspace are stored in *full storage scheme* since they do not have a special structure. In our implementation all dense matrices are stored in full storage scheme. Packed, band and rectangular full packed storage schemes exist and most linear algebra libraries deal with these different schemes. We note that the Hessenberg matrix can be stored in band storage scheme. Figures 4.2, 4.3 show how the vectors of the basis are stored in memory for both methods GMRES-MDR

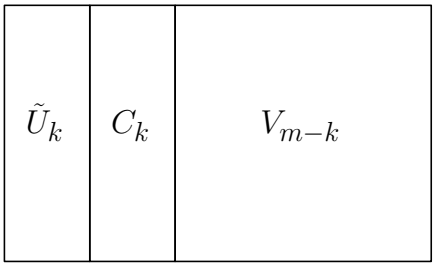

Figure 4.2: Deflation and search subspace basis vectors storage for GMRES-MDR method

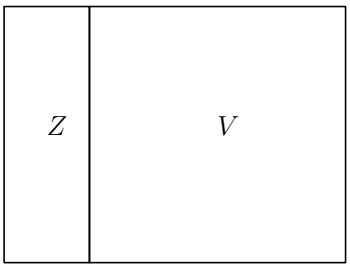

Figure 4.3: Deflation and search subspace basis vectors storage for RD-EGMRES method

and RD-EGMRES, respectively.

Sparse matrices have a special structure. In order to store a sparse matrix it is more economic to store only non-zero values. Different schemes exist, as compressed sparse row CSR, compressed sparse column CSC, coordinate COO, diagonal, skyline and block sparse row formats. Due to constraints in our application we use the CSR format. This format consists of three arrays, `val`, `colInd` and `rowPtr` of length nnz, nnz and $n+1$, respectively, where nnz is the number of non-zero elements of the matrix and n is the number of the rows of the matrix. The element j in the array `val` (resp. `colInd`) contains in the j^{th} element the value (resp. column index) of the j^{th} non-zero element of the matrix in row major style. The array `rowPtr` contains in element j the index of the element in `val` that is the first non-zero element in the j^{th} row. The last element of `rowPtr` is the number of non-zero elements of the matrix. The following example illustrates the CSR storage scheme.

$$A = \begin{pmatrix} 1 & & & -3 \\ & -9 & & \\ & & 3 & 1 \\ & & 6 & \end{pmatrix}.$$

The three corresponding arrays are given as

- `val` $= \{1, -3, -9, 3, 1, 6\}$
- `colInd` $= \{0, 3, 1, 2, 3, 2\}$

- `rowPtr` = {0, 2, 3, 5, 6}

4.1.2 Parallel interaction environment and implementation language

We use message passing interface MPI [67] for communication between processors. We use OpenMP for multithreading. These two libraries are compatible with the language C that we chose to use.

4.1.3 BLAS

The BLAS (Basic Linear Algebra Subprograms) are routines that provide standard building blocks for performing basic vector and matrix operations. The Level 1 BLAS performs scalar, vector and vector-vector operations, the Level 2 BLAS performs matrix-vector operations, and the Level 3 BLAS performs matrix-matrix operations. Because the BLAS are efficient, portable, and widely available, they are commonly used in the development of high quality linear algebra software, Linear Algebra Package (LAPACK) for example.

The following routines are used in our implementation

- `ddot`: computes a vector-vector dot product $x^\top y$
- `daxpy`: computes a scalar-vector product and adds the result to a vector $y = \alpha x + \beta y$
- `dgemv`: computes a scalar-matrix-vector product $y = \alpha Ax + \beta y$
- `dgemm`: computes a scalar-matrix-matrix product $C = \alpha AB + \beta C$
- `dtrsv`: triangular solver for one right-hand side $Tx = b$
- `dtrsm`: triangular solver for multi-right-hand sides $TX = B$
- `dnrm2`: computes a 2-norm of a vector $\alpha = \sqrt{x^\top x}$
- `drotg`: setups Givens rotation
- `drot`: applies Givens rotation

4.1.4 Sparse BLAS

Sparse BLAS Level 2 is a group of routines and functions that perform operations between a sparse matrix and dense vectors. Sparse BLAS Level 3 is a group of routines and functions that perform operations between a sparse matrix and dense matrices.

The following routines are used in our implementation:

- `dcsrmv`: computes matrix-vector product of a sparse general matrix in the CSR format $y = \alpha Ax + \beta y$
- `dcsrmm`: computes matrix-vector product of a sparse general matrix in the CSR format $Y = \alpha AX + \beta Y$

- `dcsrsv`: triangular solver for a sparse matrix in the CSR format and one right-hand side $Tx = b$
- `dcsrsm`: triangular solver for a sparse matrix in the CSR format and multi-right-hand sides $TX = B$

4.1.5 LAPACK

Linear Algebra Package is a group of routines that solve linear systems of equations, least-squares problems, eigenvalue problems and singular value problems. In the following we present the routines that we use in our implementation

Linear equations

- `dpotrf`: computes a Cholseky factorization of an SPD matrix $A = R^\top R$
- `dtrtrs`: solves a system of linear equations with a triangular coefficient matrix, with multiple right-hand sides $TX = B$

QR factorization

- `dorgqr`: computes a QR factorization of a rectangular matrix $A = QR$
- `dormqr`: multiplies a real matrix by the orthogonal matrix Q of the QR factorization formed by dgeqrf
- `dgeqrt`: computes a blocked QR factorization of a rectangular matrix $A = QR$
- `dgemqrt`: multiplies a real matrix by the orthogonal matrix Q of the blocked QR factorization formed by dgeqrt
- `dgeqp3`: computes the QR factorization of a general matrix with column pivoting (Rank Revealing QR) using level 3 BLAS

Simple non-symmetric eignevalues and singular value problems

- `dgesdd`: computes the singular value decomposition of a general rectangular matrix
- `dgebal`: balances a general matrix to improve the accuracy of computed eigenvalues and eigenvectors
- `dgebak`: transforms eigenvectors of a balanced matrix to those of the original non-symmetric matrix
- `dhseqr`: computes eigenvalues and the Schur complement of a Hessenberg matrix using the QR algorithm
- `dtrexc`: reorders the Schur decomposition of a general matrix

Symmetric generalized eigenvalue problem

- `dsygst`: reduces a real symmetric-definite generalized eigenvalue problem to the standard form
- `dsytrd`: reduces a real symmetric matrix to tridiagonal form
- `dorgtr`: generates the real orthogonal matrix Q determined by dsytrd
- `dsteqr`: computes all eigenvalues and eigenvectors of a symmetric or Hermitian matrix reduced to tridiagonal form

Nonsymmetric generalized eigenvalues problem

- `dgghrd`: reduces a pair of matrices to generalized upper Hessenberg form using orthogonal transformations
- `dggbal`: balances a pair of general real or complex matrices
- `dgghd3`: reduces a pair of matrices to generalized upper Hessenberg form
- `dhgeqz`: implements the QZ method for finding the generalized eigenvalues of the matrix pair
- `dtgexc`: reorders the generalized Schur decomposition of a pair of matrices (A, B) so that one diagonal block of (A, B) moves to another row index
- `dormhr`: multiplies a matrix C by the orthogonal matrix Q that has been determined by a preceding call to dgehrd
- `dggbak`: forms the right or left eigenvectors of a generalized eigenvalue problem by backward transformation on the computed eigenvectors of the balanced pair of matrices output by dggbal

The Intel(R) Math Kernel Library [38] offers an implementation of the previous routines that is optimized for Intel processors. Since both methods, RD-EGMRES and GMRES-MDR, do not need an explicit coefficient matrix, we use a reverse communication interface (RCI) [19]. Algorithms 20 presents the schemes of the RCI loop used by both methods.

Algorithms 21 present the schemes of the RCI routine.

4.1.6 Preconditioner and parallel matrix-vector multiplication

In our implementation we provide a block Jacobi preconditioner with two types of factorization, namely an LU and an Incomplete-LU with 0 fill-in, $ILU(0)$ [60]. These two types are obtained by calling the corresponding MKL routines after extracting the block diagonal matrix. The LU solver is the Intel MKL PARDISO. The scheme of the parallel sparse matrix-matrix and sparse matrix-vector multiplication is presented in Algorithm 22

Algorithm 20 RCI loop

```
1: ido = 0
2: while ido != 99 do
3:    call the reverse communication interface routine 21 and get the value of ido
4:    if ido == 1 then
5:       apply the preconditioned operator
6:    end if
7:    if ido == 2 then
8:       apply the preconditioner
9:    end if
10: end while
```

4.2 RD-EGMRES solver

In this section we discuss the deflation preconditioner and their impact on the synchronization steps in RD-EGMRES. As we will see, for the first sight, using a left deflation preconditioner will add an extra global communication. However, by performing a reordering of the mathematical operations, we bring back the original communication cost of the method. An extra local computation would be the price to avoid the extra communication.

4.2.1 Left and right preconditioning

GMRES-like methods minimize the norm of the residual corresponding to the preconditioned system. Let (4.1), be the linear system that we want to solve.

$$Ax = b. \tag{4.1}$$

Let M be a preconditioner of A. The left and the right preconditioned linear system are given in (4.2) and (4.3), respectively.

$$M^{-1}Ax = M^{-1}b. \tag{4.2}$$

$$AM^{-1}x = b. \tag{4.3}$$

Hence, the GMRES-like method would not minimize the same quantity in both the variants. The right preconditioned GMRES-like method minimizes the norm of the residual $r = b - Ax$ over the Krylov subspace $K(AM^{-1}, b)$. However, in the left preconditioned GMRES-like, the quantity to be minimized is the norm of the preconditioned residual $M^{-1}r$. For this reason, it is more preferable to use the right preconditioned variant of GMRES-like methods.

Algorithm 21 RCI routine

Require: ido the interface indicator, parameters of the solver

static variable $status = 0$
if $ido == 0$ **then**
  set up the solver using the parameters
  if deflation vectors exist **then**
    set $ido = 1$, $status = -1$
    return to apply the preconditioned operator on the deflation vectors
  else
    $status = 1$
  end if
end if
if $status = -1$ **then**
  prepare the deflation vectors and matrices
  set $ido = 10$, $status = 1$
  return
end if
if $status = 1$ **then**
  prepare for the Arnoldi procedure
  set $ido = 1$, $status = 2$ and return
end if
if $status = 2$ **then**
  orthogonalization of new basis vector and Hessenberg operations and stopping criterion
  if convergence and max interior iteration are not achieved **then**
    $ido = 1$, $status = 2$
    return to apply the preconditioned operator on the last basis vector
  else
    solve the least squares problem and recover the solution
    compute the deflation subspace
    if convergence or the max outer iteration is achieved **then**
      $ido = 2$, $status = 99$
      return to recover the unpreconditioned solution
    else if max interior iteration is achieved **then**
      $ido = 1$, $status = 3$
      return to apply the preconditioned operator on the solution
    end if
  end if
end if
if $status = 3$ **then**
  compute the residual
  set $ido = 10$, $status = 1$ and return
end if
if $status = 99$ **then**
  set $ido = 99$, $status = 0$ and return
end if

Algorithm 22 Parallel SpMM

Require: Δ the set of destinations neighbours processors, Γ the set of depending neighbour processors, A the matrix given in block matrices A_j for $j \in \Gamma$ and the block diagonal A_D, V a vector or a set of vectors
Ensure: compute $W = AV$ in parallel
1: **for** $j \in \Gamma$ **do**
2: nonblocking receive X_j from processor j
3: **end for**
4: **for** $j \in \Delta$ **do**
5: nonblocking send V to processor j
6: **end for**
7: compute $W = A_D V$
8: **for** $j \in \Gamma$ **do**
9: compute $W = W + A_j X_j$
10: **end for**

4.2.2 Deflation correction

We saw in Chapters 3 and 2 how the deflation subspace correction can modify the spectrum of the matrix. This makes the spectrum of the preconditioned matrix more clustered and lets the Krylov method converge faster.

The application of the deflation correction preconditioner on a vector consists of three steps: 1- project the vector on the deflation subspace 2- perform a correction by solving a coarse space problem 3- expand this correction to the vector. The communication part is the first step which stands for a dot product operation, i.e., global communication. Algorithm 23 presents a cycle of the block Arnoldi procedure in the preconditioned by deflation BGMRES method. In both algorithms, Algorithm 23 and Algorithm 24, the keyword *Local (resp. Parallel)* means that the line operations are performed without (resp. with) communication in a parallel environment.

It is, of course, a way to maintain the efficiency in the restarted variant. Nevertheless, it would be more appropriate without the communication cost. In Algorithm 24, we present an equivalent to Algorithm 23 where the global communication related to the deflation correction is performed simultaneously with the global communication that is necessary to QR factor the basis block of vectors.

4.2.3 Structures of the solver

In this part we present the solver structures in the implemented algorithm. We define four types related to preconditioning, inexact breakdown detection, deflation, and orthogonalization strategy. They are listed below.

```
/* Preconditioner */
typedef enum {
  Prec,
```

Algorithm 23 Parallel block Arnoldi procedure with deflation correction preconditioner

Require: The matrix A, the deflation vectors Z (unitary matrix), the deflation operator T, starting set of s linearly independent vectors P_1, number of iterations m

Ensure: The basis vectors $\mathcal{V}_{m+1}$, the block Hessenberg H_m

1: Local, $S = P_1^H P_1$
2: Parallel, Allreduce sum of S
3: Local, Cholesky factorization of $S = R^H R$
4: Local, $V_1 = V_1 R^{-1}$
5: **for** $j = 1 : m$ **do**
6: Local, $B = Z^H V_j$
7: Parallel, Allreduce sum of B
8: Local, $W = V_j + Z(T^{-1} - I)B$
9: Parallel, apply Algorithm 22 to compute AW and assign the result to W
10: Local, $L = \mathcal{V}_j^H W$, where $\mathcal{V}_j = \{V_1, \ldots, V_j\}$
11: Local, $W = W - \mathcal{V}_j L$
12: Local, concatenate L to the right of the block Hessenberg H_{j-1}
13: Local, $S = W^H W$
14: Parallel, Allreduce sum of S
15: Local, Cholesky factorization of $S = R^H R$
16: Local, concatenate $(0_{s,(j-1)s}, R)$ to the bottom of H_{j-1}
17: Local, $V_{j+1} = W R^{-1}$
18: **end for**

Algorithm 24 Parallel block Arnoldi procedure with deflation correction preconditioner-reordered version

Require: The matrix A, the deflation vectors Z (unitary matrix), the deflation operator T, starting set of s linearly independent vectors P_1, number of iterations m
Ensure: The basis vectors $\mathcal{V}_{m+1}$, the block Hessenberg H_m
1: Local, $S = P_1^H P_1$, $B = Z^H P_1$
2: Parallel, Allreduce sum of S, and B
3: Local, Cholesky factorization of $S = R^H R$
4: Local, $V_1 = V_1 R^{-1}$, $B = BR^{-1}$
5: **for** $j = 1 : m$ **do**
6: Local, $W = V_j + Z(T^{-1} - I)B$
7: Parallel, apply Algorithm 22 to compute AW and assign the result to W
8: Local, $L = \mathcal{V}_j^H W$, where $\mathcal{V}_j = \{V_1, \ldots, V_j\}$
9: Local, $W = W - \mathcal{V}_j L$
10: Local, concatenate L to the right of the block Hessenberg H_{j-1}
11: Local, $S = W^H W$, $B = Z^H W$
12: Parallel, Allreduce sum of S, and B
13: Local, Cholesky factorization of $S = R^H R$
14: Local, concatenate $(0_{s,(j-1)s}, R)$ to the bottom of H_{j-1}
15: Local, $V_{j+1} = WR^{-1}$, $B = BR^{-1}$
16: **end for**

```
  NoPrec
} Prec_t;
```

`Prec_t` is a type defining if a preconditioner is to be used.

```
/* Inexact breakdown criterion */
typedef enum {
  RRQR,
  SVD,
  NoRed
} BlcRed_t;
```

`BlcRed_t` is a type defining the criterion test to detect the inexact breakdown, `RRQR` (resp. `SVD`) applies the detection of inexact breakdown using a criterion based on the rank revealing QR (resp. singular value decomposition) of the block residual, see Section 2.4.1 for more details. `NoRed` desactivates the inexact breakdown detection.

```
/* Deflation */
typedef enum {
  Def,
  NoDef
} Def_t;
```

`Def_t` is a type that defines if deflation would be used or not.

```
/* Orthogonalization type */
```

```
typedef enum {
  BCGS,
  DBCGS
} Ortho_t;
```

`Ortho_t` is a type that defines which strategy of orthogonalization the solver uses. Two choices are available, `BCGS` that stands for block classical Gram-Schmidt or `DBCGS` that stands for double block classical Gram-Schmidt.

In order to measure time we provide a structure for timing.

```
/* Timing structure */
typedef struct{
  double TotalTime;                /** Total time of the solve                    */
  double PreconditioningTime;      /** Time consumed in preconditioning           */

  double HessenbergOperationsTime; /** Time consumed in Hessenberg QR update
                                       (without inexact breakdown detection time) */
  double DeflationComputationTime; /** Time in solving EVP in KS, preparing
                                       T matrix, expansion of vectors             */
  double DeflationComputationComm; /** Time in orthogonalization of new
                                       deflated vectors and reduction of Z^T A Z  */
  double DeflationApplicationTime; /** Time in applying deflation operator
                                       on basis vectors disregarding communication */
  double OrthoAndDefAsynch;        /** Time in reduction to apply the deflation
                                       operator on basis vectors                  */
  double SPMMTime;                 /** Time in sparse matrix matrix multiplication */
  double SPMMDiag;                 /** Partial time of sparse matrix matrix
                                       multiplication (diagonal block)            */
  double SPMMOffDiag;              /** Partial time of sparse matrix matrix
                                       multiplication (off-diagonal block)        */
  double SPMMComm;                 /** Partial time of sparse matrix matrix
                                       multiplication (communication)             */
  double OrthogonalizationTime;    /** Partial time in ortho. (computation)       */
  double OrthogonalizationComm;    /** Partial time in ortho. (communication)     */
  double InexactBreakDownTime;     /** Inexact breakdown computation time         */
  double Recovering;               /** Time consumed in recovering sol. & res     */
} Timing_t;
```

The main structure of the method contains the elements that are used in the algorithm and the parameters of the solver.

```
typedef struct{
  /** Basis*/
  double*   V;             /** Basis of Enlarged Krylov SS */
  int*      VIdx;          /** Indices of block vector*/

  /** Deflation*/
  double*   Z;             /** Deflation basis*/
  int       dimZ;          /** Deflation dimension*/
  double*   WorkZ;         /** Work place for deflation*/
  double*   T;             /** Factored Deflation matrix*/
  double*   CopyT;         /** Deflation matrix*/
  int*      TLUP;          /** Permutation of LU factorization*/
  int       MaxDefDim;     /** Maximal dimension of deflation subspace*/
```

```
/** Hessenberg*/
double*   H;              /** Hessenberg Matrix*/
double*   Htau;           /** Hessenberg Householder coeff*/
double*   H2;             /** Hessenberg Matrix*/

/** Breakdown**/
double*   Q;              /** Breakdowns Roations*/
double*   Qtau;           /** Breakdowns Roations Householder coeff*/
int*      BIdx;           /** Breakdown Indices*/
int*      BSize;          /** Block size over iterations*/

/** Shared variable*/
double*   EKRHS;          /** Enlarged Krylov RHS*/
double*   EKSolution;     /** Enlarged Krylov Solution*/
double*   KRHS;           /** Kylov RHS*/
double*   rvec;           /** Vector containing residual norm of iterations*/

/** System vectors*/
double*   b;              /** right hand side*/
double*   Solution;       /** Solution to be returned*/
double*   Residual;       /** Residual to be returned*/
int       ln;             /** Local number of degrees of freedom*/
int       ldv;            /** Local leading dimension of V*/

/** Parameters for the solver*/
int      MaxBasis;        /** Max dimension of enlarged Krylov subspace*/
int      EF;              /** Enlarging Factor*/
int      ND;              /** N° of domains*/
int      s;               /** N° of vectors on which the user applies his operator
                              or preconditioner in the RCI*/
int      MaxDefPerCycle;  /** Maximal number of eigenvalues deflated during a cycle*/
int      iteration;       /** Actual number of interior iteration*/
int      Cycle;           /** Actual number of exterior iteration*/
int      MaxCycle;        /** Maximum number of outer iterations*/
int      GIter;           /** Static (over restarts) iteration*/
int      ActualDefDim;    /** Actual dimension of deflation subspace*/
int      ActualSize;      /** Actual dimension of the block*/
int      IBreakdown;      /** N° of inexact breakdown*/
double   normb;           /** norm of the rhs*/
double   Ctol;            /** Convergence threshold*/
double   EVthreshold;     /** Eigenvalue deflation threshold*/
Ortho_t  Ortho;           /** Orthogonalization strategy*/
BlcRed_t Red;             /** Inexact breakdown strategy*/
Prec_t   Prec;            /** Flag for using a preconditioner*/
Def_t    Def;             /** Flag for using deflation of eigenvalues*/
MPI_Comm comm;            /** Communicator*/
int      rank;            /** Rank of proc*/
int      AsyncDef;        /** Flag to indicate if an asynchronous deflation is done*/

/** Arrays*/
double*   EResidual;      /** Enlarged residual*/
double*   ESolution;      /** Enlarged solution*/
```

```
  double*   WorkV;          /** Work space for basis vectors*/
  double*   WorkH;          /** Work space for Enlarged Krylov subspace*/
  double*   U;              /** Pointer to the matrix on which we apply the operator*/
  double*   AU;             /** Pointer to the matrix that results from applying
                                the operator on U (previous parameter) */
  Timing_t  Timing;         /** Structure for timings*/
}EGMRES_t;
```

4.3 GMRES-MDR solver

In this section we present the C structs that we used to implement the GMRES-MDR method.

We define three types related to preconditioning, deflation strategy, and orthogonalization strategy. They are listed below.

```
/* Preconditioner */
typedef enum {
  Prec,
  NoPrec
} Prec_t;
```

`Prec_t` is a type defining if a preconditioner is to be used.

```
/* Deflation */
typedef enum {
  RITZ,
  HRITZ,
  SVD,
  NoDef
} Def_t;
```

`Def_t` is a type that defines which strategy of deflation to use: `RITZ` for the Ritz pairs of the matrix A, `HRITZ` for the Ritz pairs of the matrix A^{-1} (harmonic Ritz), `SVD` for the Ritz pairs of the matrix $A^H A$, and `NoDef` which stands for no deflation strategy to be used.

```
/* Orthogonalization type */
typedef enum {
  BCGS,
  DBCGS
} Ortho_t;
```

`Ortho_t` is a type that defines which strategy of orthogonalization the solver will use. Two choices are available, `BCGS` that stands for block classical Gram-Schmidt or `DBCGS` that stands for double block classical Gram-Schmidt.

In order to measure time we provide a structure for timing.

```
/* Timing structure */
typedef struct{
  double TotalTime;               /** Total time of the GMRESMDR method           */
  double PreconditioningTime;     /** Time for application of
```

```
                                    the preconditioner                    */
  double HessenbergOperationsTime; /** Time of Hessneberg operations       */
  double DeflationComputationTime; /** Time of computing deflation
                                    without communication                 */
  double DeflationComputationComm; /** Time of communication necessary
                                    in deflation                          */
  double SPMVTime;                 /** Time for Sparse matrix vector
                                    multiplication                        */
  double SPMVDiag;                 /** Time for Diagonal operator SPMM(V)  */
  double SPMVOffDiag;              /** Time for Off-Diagonal operator SPMM(V) */
  double SPMVComm;                 /** Time of communication in SPMM(V)
                                    multiplication                        */
  double OrthogonalizationTime;    /** Time of orthogonalization           */
  double OrthogonalizationComm;    /** Time of communication in
                                    orthogonalization                     */
  double Recovering;               /** Time of recovering solution
                                    and residual                          */
} Timing_t;
```

The main structure of the method contains the elements that are used in the algorithm and the parameters of the solver.

```
typedef struct{
  /** Basis                                                              */
  double*  V;            /** Basis of Enlarged Krylov SS                 */
  int      dimV;         /** Dimension of V                              */

  /** Deflation                                                          */
  double*  Z;            /** Deflation basis                             */
  double*  AZ;           /** A * Deflation basis                         */
  double*  WorkZ;        /** Work space for Deflation basis              */
  int      dimZ;         /** Deflation dimension                         */
  int      MaxDefDim;    /** Maximal dimension of deflation subspace     */

  /** Hessenberg                                                         */
  double*  H;            /** Hessenberg Matrix                           */
  double*  Htau;         /** Hessenberg Householder coeff                */
  double*  H2;           /** Hessenberg Matrix                           */
  double*  GtG;          /** Matrix A in generalized EVP                 */
  double*  VtV;          /** Matrix B in the generalized EVP             */
  double*  VtW;          /** Factor of matrix A in the generalized EVP   */
  double*  ZtZ;          /** Part of matrix B in the generalized EVP     */
  double*  D;            /** Diagonal matrix in the Hessenberg           */

  /** Shared variables                                                   */
  double*  KRHS;         /** Kylov RHS                                   */
  double*  KSOL;         /** Kylov SOL                                   */
  double*  rvec;         /** Vector containing history of
                                    residual norm                         */

  /** System vectors                                                     */
  double*  b;            /** right hand side                             */
  double*  Solution;     /** Solution to be returned                     */
```

```
  double*  Residual;     /** Residual to be returned                  */
  int      ln;           /** Local number of degrees of freedom       */
  int      ldv;          /** Local leading dimension of V             */

  /** Parameters for the solver                                       */
  int      MaxBasis;     /** Max dimension of enlarged Krylov subspace */
  int      ND;           /** Number of domains                        */
  int      s;            /** Number of vectors on which the user
                             applies his operator or preconditioner   */
  int      iteration;    /** Actual number of inner iteration         */
  int      Cycle;        /** Actual number of outerr iteration        */
  int      MaxCycle;     /** Maximum number of outer iterations       */
  int      GIter;        /** Static (over restarts) iteration         */
  double   normb;        /** norm of the rhs                          */
  double   Ctol;         /** Convergence threshold                    */
  Ortho_t  Ortho;        /** Orthogonalization strategy               */
  Prec_t   Prec;         /** Flag for using a preconditioner          */
  Def_t    Def;          /** Flag for using deflation of eigenvalues  */
  MPI_Comm comm;         /** Communicator                             */
  int      rank;         /** Rank of proc                             */

  /** Arrays                                                          */
  double*  WorkV;        /** Work space for basis vectors             */
  double*  WorkH;        /** Work space for Krylov subspace           */

  double*  U;            /** Pointer to the matrix on which we
                             apply the operator                       */
  double*  AU;           /** Pointer to the matrix that results
                             from applying the operator on U
                             (previous parameter)                     */
  Timing_t Timing;       /** Structure for timings                    */
}GMRESMDR_t;
```

4.4 Parallel numerical experiments

In this section we present the numerical experiments of our implementation of the methods RD-EGMRES and GMRES-MDR.

All experiments are performed on the supercomputer PANGEA at TOTAL. We had access to a maximum of 1024 nodes. Each node has 16 cores. Our set of test matrices consists of the matrices SPE10 and Grid2D-2000x2000. We solve two sequences, each of nine linear systems in which the matrix is fixed. The right-hand sides are different for each linear system in the sequence. All variants have the same right-hand side for the i^{th} linear system, $i = 1,\ldots,9$. The matrix is SPE10 $\in \mathbb{R}^{n\times n}$ where n is the number of unknowns $n = 1094421$, the number of the non-zero elements $nnz = 7478141$. SPE10 corresponds to a simulation of a black oil with a heterogeneous permeability field [14]. The grid has 1094421 active cells. The matrix Grid2D-2000x2000 $\in \mathbb{R}^{n\times n}$, where n is the number of unknowns $n = 4\times 10^6$, the number of non-zero elements $nnz = 2\times 10^7$. This matrix corresponds to a discretization of a pressure system in which the permeability field follows a log-normal distribution with standard deviation $\sigma = 3$ and a mean

$\mu = log(500)$. The discretized domain is the unit square with 2000×2000 grid points. In all experiments, a linear system is solved to warm up the machine before solving each sequence.

Figure 4.4 presents a scalability curve of the method RD-EGMRES. The sequence of linear systems are related to the test matrix SPE10. The number of cores varies from 64 up to 4096 (4 cores per MPI processor). A block Jacobi preconditioner with LU factorization in each block is used. The number of blocks is equal to the number of processors. We remark that the method RD-EGMRES scales well by increasing the number of processors.

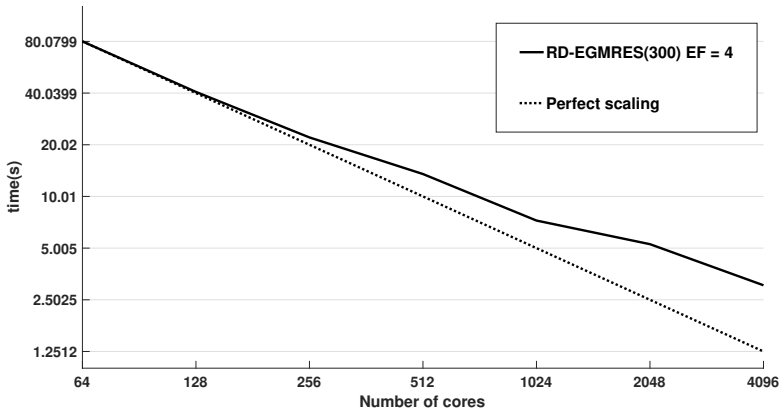

Figure 4.4: Scalability curve of RD-EGMRES(300) with enlarging factor of 4. Sequence of nine linear systems related to the matrix, SPE10, with different right-hand sides. 4 cores per MPI processor. Preconditioner: block Jacobi preconditioner with LU factorization in the block. The dashed line presents the perfect scalability curve.

In Listing 4.1 and Listing 4.2, we present two post-analysis of RD-EGMRES(300) that is associated to Figure 4.4.

Figure 4.5 shows the scalability of the three variants discussed in Section 3.4. The sequence of linear systems is related to the test matrix Grid2D-2000x2000. The number of nodes (processors) varies from 16 up to 1024 with 2 cores per processor. A block Jacobi preconditioner with LU factorization in each block is used. The number of blocks is equal to the number of processors. In Listing 4.3 and Listing 4.4, we present two post-analysis of GMRES-MDR with SVD deflation variant that is associated to Figure 4.5.

Figure 4.6 shows the scalability of the three variants discussed in Section 3.4. The sequence of linear systems is related to the test matrix SPE10. The number of nodes (processors) varies from 16 up to 256, 1 core per MPI processor. A block Jacobi preconditioner with $ILU(0)$ factorization in each block is used. The number of blocks is equal to the number of processors.

We notice that the three variants scale well with a smaller runtime for the SVD variant. We note that the runtime for the three variants of deflation does not scale

Listing 4.1: Post analysis of RD-EGMRES(300). First linear system in the sequence SPE10.

```
Solver information:
Number of processors: 1024
Number of threads: 4
Enlarging factor: 4
Preconditioner: LU of block Jacobi
Orthogonalization strategy: Double Block Classical Gram-Schmidt
Max dimension of search subspace: 300
Threshold for deflation: 1.00000000000000e-02
--------------------------------------------------------------------------------
Post analysis:
--------------------------------------------------------------------------------
Number of iterations: 226
Number of cycles: 3
Residual norm: 9.78227506437520e-02
Relative residual norm: 9.78227506437520e-02
Relative tolerance of convergence: 1.00000000000000e-01
Number of deflated eigenvalues: 26
--------------------------------------------------------------------------------
Times over processors          :      MAX       |   MIN       |   AVERAGE
  Total                        :     1.252737   |  1.252737   |  1.252737 s
  Preconditioning              :     0.597092   |  0.323088   |  0.377136 s
  SPMM                         :     0.335039   |  0.020960   |  0.091370 s
  SPMM Diag                    :     0.003180   |  0.002491   |  0.002608 s
  SPMM Off Diag                :     0.016838   |  0.003171   |  0.009045 s
  SPMM Comm                    :     0.289093   |  0.013585   |  0.072732 s
  Deflation Computation        :     0.230229   |  0.166689   |  0.195734 s
  Deflation Computation comm   :     0.038693   |  0.000659   |  0.037440 s
  DeflationApplication         :     0.007838   |  0.006816   |  0.007116 s
  DeflationApplication comm    :     0.025552   |  0.013386   |  0.020000 s
  Orthogonalization            :     0.054282   |  0.041502   |  0.044882 s
  Orthogonalization comm       :     0.488362   |  0.205556   |  0.427135 s
  InexactBreakDown             :     0.042400   |  0.025673   |  0.026436 s
  HessenbergOperations         :     0.029802   |  0.018241   |  0.018845 s
--------------------------------------------------------------------------------
```

Listing 4.2: Post analysis of EGMRES. Ninth linear system in the sequence SPE10.

```
Solver information:
Number of processors: 1024
Number of threads: 4
Enlarging factor: 4
Preconditioner: LU of block Jacobi
Orthogonalization strategy: Double Block Classical Gram-Schmidt
Max dimension of search subspace: 300
Threshold for deflation: 1.00000000000000e-02
-------------------------------------------------------------------------
Post analysis:
-------------------------------------------------------------------------
Number of iterations: 41
Number of cycles: 1
Residual norm: 9.13024960343515e-02
Relative residual norm: 9.13024960343515e-02
Relative tolerance of convergence: 1.00000000000000e-01
Number of deflated eigenvalues: 35
-------------------------------------------------------------------------
Times over processors          :      MAX       |    MIN       |   AVERAGE
  Total                        :     0.217336   |   0.217336   |   0.217336 s
  Preconditioning              :     0.126769   |   0.059978   |   0.069642 s
  SPMM                         :     0.066835   |   0.002928   |   0.012515 s
  SPMM Diag                    :     0.000694   |   0.000434   |   0.000470 s
  SPMM Off Diag                :     0.002844   |   0.000562   |   0.001591 s
  SPMM Comm                    :     0.058184   |   0.001693   |   0.009676 s
  Deflation Computation        :     0.022635   |   0.022460   |   0.022548 s
  Deflation Computation comm   :     0.000000   |   0.000000   |   0.000000 s
  DeflationApplication         :     0.002531   |   0.002075   |   0.002185 s
  DeflationApplication comm    :     0.013028   |   0.009607   |   0.012503 s
  Orthogonalization            :     0.007747   |   0.005684   |   0.006122 s
  Orthogonalization comm       :     0.092913   |   0.032998   |   0.085236 s
  InexactBreakDown             :     0.004346   |   0.002770   |   0.002867 s
  HessenbergOperations         :     0.003434   |   0.002030   |   0.002107 s
  RecoveringOperations         :     0.000457   |   0.000255   |   0.000369 s
-------------------------------------------------------------------------
```

Listing 4.3: Post analysis of GMRES-MDR with SVD deflation variant. First linear system in the sequence Grid2D-2000x2000.

```
Solver information:
Number of processors: 1024
Number of threads: 2
Preconditioner: LU of block Jacobi
Orthogonalization strategy: Double Block Classical Gram-Schmidt
Deflation strategy: SVD values
Max dimension of search subspace: 60
Max dimension of deflation subspace: 20
-------------------------------------------------------------------------
Post analysis:
-------------------------------------------------------------------------
Number of iterations: 1395
Number of cycles: 36
Residual norm: 9.96438063135672e-02
Relative residual norm: 9.96438063135672e-02
Relative tolerance of convergence: 1.00000000000000e-01
-------------------------------------------------------------------------
Times over processors     :      MAX       |   MIN       |   AVERAGE
  Total                   :     3.237501   |  3.237501   |  3.237501 s
  Preconditioning         :     1.577664   |  1.209435   |  1.294308 s
  SPMV                    :     0.447073   |  0.081644   |  0.166228 s
  SPMV Diag               :     0.031328   |  0.025887   |  0.026805 s
  SPMV Off Diag           :     0.068317   |  0.016273   |  0.044505 s
  SPMV Comm               :     0.322190   |  0.036086   |  0.090636 s
  DeflationComputation    :     0.280072   |  0.275017   |  0.276312 s
  DeflationComm           :     0.021925   |  0.016195   |  0.020721 s
  Orthogonalization       :     1.539758   |  1.132159   |  1.445061 s
  Orthogonalization Comm  :     1.094884   |  0.642479   |  0.984363 s
  HessenbergOperations    :     0.002427   |  0.002085   |  0.002221 s
-------------------------------------------------------------------------
```

Listing 4.4: Post analysis of GMRES-MDR with SVD deflation variant. Ninth linear system in the sequence Grid2D-2000x2000.

```
Solver information:
Number of processors: 1024
Number of threads: 2
Preconditioner: LU of block Jacobi
Orthogonalization strategy: Double Block Classical Gram-Schmidt
Deflation strategy: SVD values
Max dimension of search subspace: 60
Max dimension of deflation subspace: 20
-------------------------------------------------------------------------
Post analysis:
-------------------------------------------------------------------------
Number of iterations: 39
Number of cycles: 1
Residual norm: 9.99391435927701e-02
Relative residual norm: 9.99391435927701e-02
Relative tolerance of convergence: 1.00000000000000e-01
-------------------------------------------------------------------------
Times over processors     :      MAX       |   MIN       |   AVERAGE
  Total                   :      0.069219  |   0.069219  |   0.069219 s
  Preconditioning         :      0.038246  |   0.034066  |   0.036175 s
  SPMV                    :      0.006581  |   0.002130  |   0.004383 s
  SPMV Diag               :      0.001015  |   0.000734  |   0.000759 s
  SPMV Off Diag           :      0.001926  |   0.000461  |   0.001252 s
  SPMV Comm               :      0.004382  |   0.000921  |   0.002392 s
  DeflationComputation    :      0.003500  |   0.003277  |   0.003379 s
  DeflationComm           :      0.000360  |   0.000173  |   0.000268 s
  Orthogonalization       :      0.025600  |   0.020788  |   0.023068 s
  Orthogonalization Comm  :      0.013947  |   0.008824  |   0.011384 s
  HessenbergOperations    :      0.000074  |   0.000053  |   0.000063 s
-------------------------------------------------------------------------
```

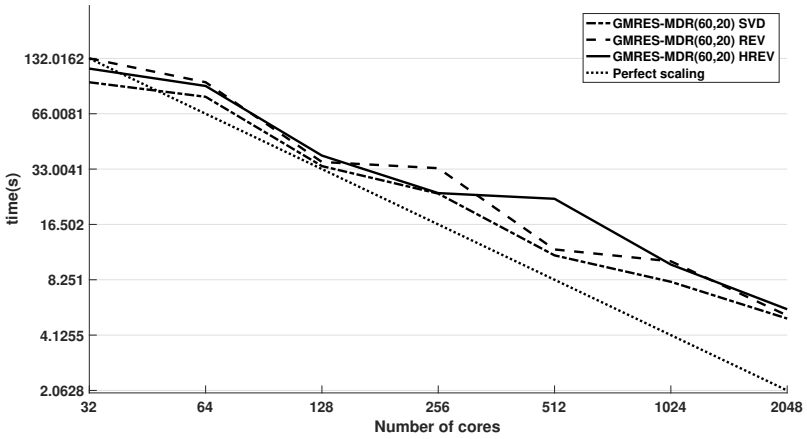

Figure 4.5: Comparison between runtime of three variants of deflation subspaces. SVD, REV and HEV stand for subspace method related to the matrices $A^H A$, A and A^{-1}, respectively. The maximum dimension of the subspace is 60. The maximum dimension of the deflation subspace is 20. Nine linear systems sharing the same matrix, Grid2D-2000x2000, with different right-hand sides (for index i the three methods share the same right-hand side).

on 512 nodes in the test case SPE10. See Figure 4.7 and the discussion below for the explanation.

Figure 4.7 shows the percentage of the average time (over all processors) spent in the operations during the solution of a representative linear system of the sequence by using the SVD deflation strategy. This runtime corresponds to a solution of the system on 256 nodes. We remark that the time spent in the computation of the generalized eigenvalue problem becomes much more important than all other operations. This explains why the scaling of the runtime stopped for this sequence after 512 nodes.

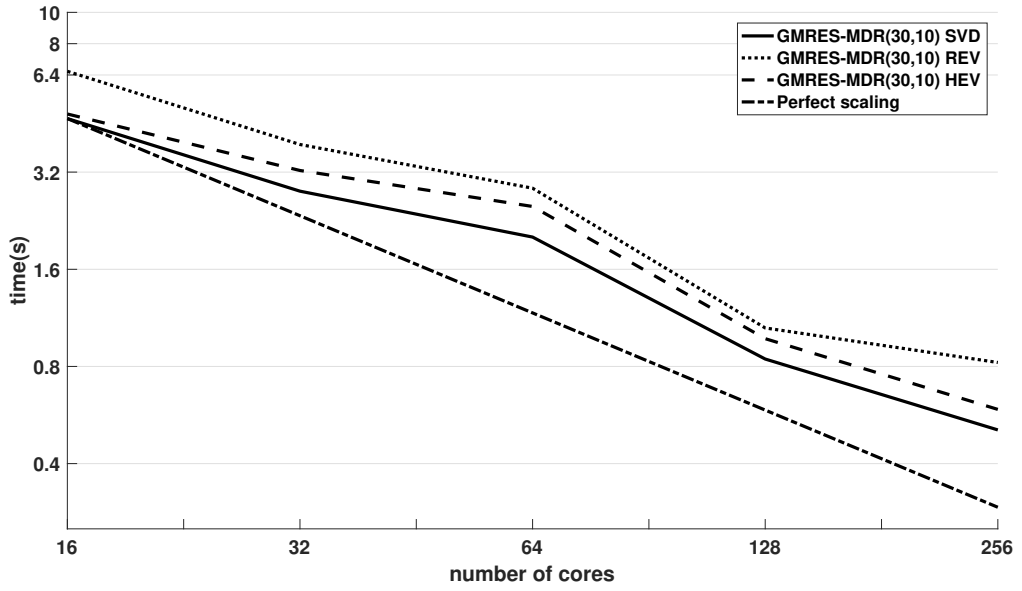

Figure 4.6: Comparison between runtime of three variants of deflation subspaces. SVD, REV and HEV stand for subspace method related to the matrices $A^H A$, A and A^{-1}, respectively. The maximum dimension of the subspace is 30. The maximum dimension of the deflation subspace is 10. Nine linear systems sharing the same matrix, SPE10, with different right-hand sides (for index i the three methods share the same right-hand side).

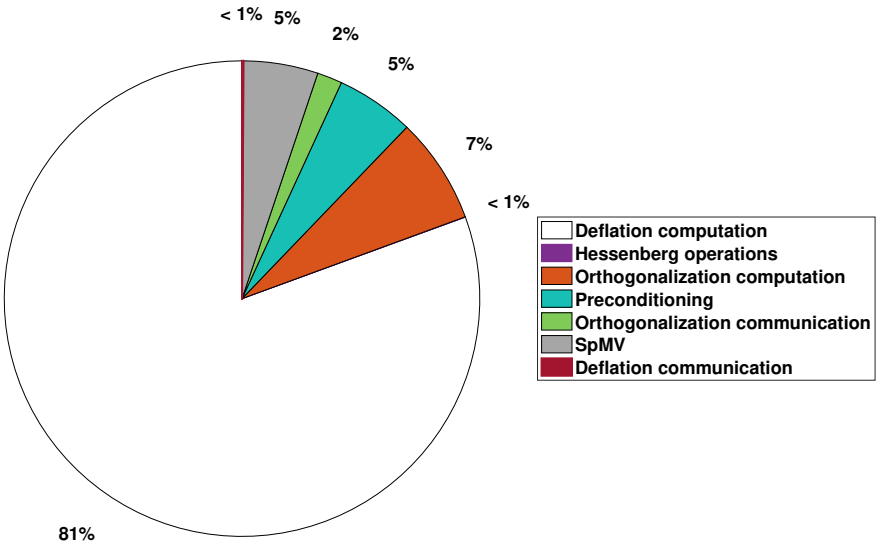

Figure 4.7: Average time (over processors) of different operations in GMRES-MDR as a percentage of the total runtime. The communication and the computation part of the deflation and the orthogonalization are separated. This runtime corresponds to the solution of a representative linear system during the sequence. Number of nodes 256. Deflation strategy is the SVD.

CHAPTER 5

ALGEBRAIC ROBUST COARSE SPACES

Outline of the current chapter

In this chapter we present a class of robust and fully algebraic two-level preconditioners for SPD matrices. We introduce the notion of algebraic local symmetric positive semi-definite (SPSD) splitting of an SPD matrix and we give a characterization of this splitting. This splitting leads to construct *algebraically and locally* a class of efficient coarse spaces which bound the spectral condition number of the preconditioned system by a number defined a priori. We also introduce the τ-filtering subspace. This concept helps compare the dimension minimality of coarse spaces. Some PDEs-dependant preconditioners correspond to a special case. The examples of the algebraic coarse spaces in this chapter

are not practical due to expensive construction. We propose a heuristic approximation that is not costly. Numerical experiments illustrate the efficiency of the proposed method.

5.1 Introduction

The conjugate gradient method CG [35] is a widely known Krylov iterative method, for solving large linear systems of equations of the form

$$Ax = b, \tag{5.1}$$

where $A \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix, $b \in \mathbb{R}^n$ is the right-hand side, and $x \in \mathbb{R}^n$ is the vector of unknowns. It finds at iteration j the approximate solution $x_j \in x_0 + K_j(A, r_0)$ that minimizes the A-norm of the error $\|x_* - x_j\|_A$, where x_0 is the initial guess, $r_0 = b - Ax_0$, $K_j(A, r_0)$ is the Krylov subspace of dimension j related to A and r_0, x_* is the exact solution of (5.1), and $\|.\|_A$ is the A-norm. The convergence of this method is well studied in the literature [60]. The rate of convergence depends on the condition number of the matrix A. Let $\kappa = \frac{\lambda_n}{\lambda_1}$ be the spectral condition number of A, where λ_n and λ_1 are the largest and the smallest eigenvalues of A respectively, the error at iteration j satisfies the following inequality

$$\|x_* - x_j\|_A \leq \|x_* - x_0\|_A \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^j. \tag{5.2}$$

We suppose that the graph of the matrix is partitioned into a number of subdomains by using a k-way partitioning method [42]. To enhance the convergence, it is common to solve the preconditioned system

$$M^{-1}Ax = M^{-1}b. \tag{5.3}$$

Block Jacobi, additive Schwarz, restricted additive Schwarz, etc., are widely used preconditioners. These preconditioners are called one-level preconditioners. They correspond to solving subproblems on subdomains. In [13, 18] the authors prove that the largest eigenvalue of the preconditioned system by the additive Schwarz preconditioner is bounded by a number that is independent of the number of subdomains. However, no control is guaranteed for the smallest eigenvalue of the preconditioned matrix. Furthermore, when the number of subdomains increases, the smallest eigenvalue might become even smaller. Thus, the number of iterations to reach convergence typically increases. This occurs since this type of preconditioner employs only local information and does not include global information. For this reason, these preconditioners are usually combined with a second-level preconditioner, which corresponds to a coarse space correction or deflation. In principle, it is meant to annihilate the impact of the smallest eigenvalues of the operator. Different strategies exist in literature to add this level. In [76], the authors compare different strategies of applying two-level preconditioners. In

[13, 77, 51, 71, 18, 30, 44], the authors propose different methods for constructing a coarse space correction. Coarse spaces can be categorized in two types, analytic and algebraic. Analytic coarse spaces depend on the underlying problem from which the matrix A is issued. Algebraic coarse spaces depend only on the coefficient matrix A and does not require information from the underlying problem from which it arises. Based on the underlying PDE and its discretization, several methods that propose analytic coarse spaces are described in literature [18, 13, 77, 51, 71].

In most cases, a generalized (or standard) eigenvalue problem is solved in each subdomain. Every subdomain then contributes to the construction of the coarse space by adding certain eigenvectors. These methods are efficient in several applications. Nevertheless, the dependence on the analytic information makes it impossible to be made in a pure algebraic way. Algebraic coarse space correction can be found in literature [30, 44]. However, the construction of the coarse space can be even more costly than solving the linear system (5.1). In this paper we discuss a class of robust preconditioners that are based on locally constructed coarse spaces. We characterize the local eigenvalue problems that allow to construct an efficient coarse space related to the additive Schwarz preconditioner. The paper is organized as follows. In 5.2 we review general theory of one- and two-level preconditioners, in 5.3 we present our main result. We introduce the notion of *algebraic local SPSD splitting of an SPD matrix*. For a simple case, given the block SPD matrix $B = \begin{pmatrix} B_{11} & B_{12} & \\ B_{21} & B_{22} & B_{23} \\ & B_{32} & B_{33} \end{pmatrix}$, the local SPSD splitting of B with respect to the first block means finding two SPSD matrices B_1, B_2 of the form $B_1 = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & * \\ & \end{pmatrix}$ and $B_2 = \begin{pmatrix} & * & B_{23} \\ & B_{32} & B_{33} \end{pmatrix}$, where $*$ represents a non-zero block matrix such that $B = B_1 + B_2$. We characterize all possible local SPSD splittings. Then we introduce the τ-filtering subspace. Given two SPSD matrices A, B, a τ-filtering subspace Z makes the following inequality hold

$$(u - Pu)^\top B(u - Pu) \leq u^\top Au, \quad \forall u,$$

where P is an orthogonal projection on Z. Based on the local SPSD splitting and the τ-filtering subspace, we propose in 5.4 an efficient coarse space, which bounds the spectral condition number by a given number defined a priori. Furthermore, we show how the coarse space can be chosen such that its dimension is minimal. The resulting spectral condition number depends on three parameters. The first parameter depends on the sparsity of the matrix, namely, the minimum number of colours k_c needed to colour subdomains such that two subdomains of the same colour are disjoint, see Lemma 10 [13, Theorem 12]. The second parameter k_m depends on the algebraic local SPSD splitting. It is bounded by the number of subdomains. For a special case of splitting it can be chosen to be the maximal number of subdomains that share a degree of freedom. The third parameter is chosen such that the spectral condition number is bounded by

the user-defined upper bound. In all stages of the construction of this coarse space, no information is necessary but the coefficient matrix A and the desired bound on the spectral condition number. We show how the coarse space constructed analytically by the method GenEO [70, 18] corresponds to a special case of our characterization. We also discuss the extreme cases of the algebraic local SPSD splitting and the corresponding coarse spaces. We explain how these two choices are expensive to construct in practice. Afterwards, we propose a practical strategy to compute efficiently an approximation of the coarse space. In 5.5 we present numerical experiments to illustrate the theoretical and practical impact of our work. At the end, we give our conclusion in 5.6.

To facilitate the comparison with GenEO we follow the presentation in [18, Chapter 7].

Notation

Let $A \in \mathbb{R}^{n\times n}$ denote a symmetric positive definite matrix. We use MATLAB notations. Let $S_1, S_2 \subset \{1,\dots,n\}$ be two sets of indices. The concatenation of S_1 and S_2 is represented by $[S_1, S_2]$. We note that the order of the concatenation is important. $A(S_1,:)$ is the submatrix of A formed by the rows whose indices belong to S_1. $A(:,S_1)$ is the submatrix of A formed by the columns whose indices belong to S_1. $A(S_1,S_2) := (A(S_1,:))(:,S_2)$. The identity matrix of size n is denoted I_n. We suppose that the graph of A is partitioned into N non-overlapping subdomains, where $N \ll n$. The coefficient matrix A is represented as $\left(a_{ij}\right)_{1\leq i,j\leq n}$. Let $\mathcal{N} = \{1,\dots,n\}$ and let $\mathcal{N}_{i,0}$ for $i \in \{1,\dots,N\}$ be the subsets of $\mathcal{N}$ such that $\mathcal{N}_{i,0}$ stands for the subset of the degrees of freedom, *DOF*, in the subdomain i. We refer to $\mathcal{N}_{i,0}$ as the *interior DOF* in the subdomain i. Let Δ_i for $i \in \{1,\dots,N\}$ be the subset of $\mathcal{N}$ that represents the neighbors DOF of the subdomain i, i.e., the DOFs of distance $= 1$ from the subdomain i through the graph of A. We refer to Δ_i as the *overlapping DOF* in the subdomain i. We denote $\mathcal{N}_i = [\mathcal{N}_{i,0},\ \Delta_i]$, $\forall i \in \{1,\dots,N\}$, the concatenation of the interior and the overlapping DOF of the subdomain i. We denote $\mathcal{C}_i$, $\forall i \in \{1,\dots,N\}$, the complementary of $\mathcal{N}_i$ in $\mathcal{N}$, i.e., $\mathcal{C}_i = \mathcal{N} \setminus \mathcal{N}_i$. We note $n_{i,0}$ the cardinality of the set $\mathcal{N}_{i,0}$, δ_i the cardinality of Δ_i and n_i the cardinality of the set $\mathcal{N}_i$, $\forall i \in \{1,\dots,N\}$. Let $R_{i,0} \in \mathbb{R}^{n_{i,0}\times n}$ be defined as $R_{i,0} = I_n(\mathcal{N}_{i,0},:)$. Let $R_{i,\delta} \in \mathbb{R}^{\delta_i\times n}$ be defined as $R_{i,\delta} = I_n(\Delta_i,:)$. Let $R_i \in \mathbb{R}^{n_i\times n}$ be defined as $R_i = I_n([\mathcal{N}_{i,0},\Delta_i],:)$. Let $R_{i,c} \in \mathbb{R}^{(n-n_i)\times n}$ be defined as $R_{i,c} = I_n(\mathcal{C}_i,:)$. Let $\mathcal{P}_i = I_n([\mathcal{N}_{i,0},\Delta_i,\mathcal{C}_i],:) \in \mathbb{R}^{n\times n}$, be a permutation matrix associated to the subdomain i, $\forall i \in \{1,\dots,N\}$. We denote $D_i \in \mathbb{R}^{n_i,\times n_i}$, $i = 1,\dots,N$, any non-negative diagonal matrix such that

$$I_n = \sum_{i=1}^{N} R_i^\top D_i R_i. \tag{5.4}$$

We refer to $(D_i)_{1\leq i\leq N}$ as the algebraic partition of unity. Let n_0 be a positive integer, $n_0 \ll n$. Let $V_0 \in \mathbb{R}^{n\times n_0}$ be a tall and skinny matrix of full rank. We denote $\mathcal{S}$ the subspace generated by the columns of V_0. This subspace will stand for the coarse space. We denote R_0 the projection operator on $\mathcal{S}$. We denote $R_0^\top$ the interpolation operator

from $\mathcal{S}$ to the global space. Let $\mathscr{R}_1$ be the operator defined by:

$$\begin{aligned} \mathscr{R}_1 : \prod_{i=1}^{N} \mathbb{R}^{n_i} &\to \mathbb{R}^n \\ (u_i)_{1\le i\le N} &\mapsto \sum_{i=1}^{N} R_i^\top u_i. \end{aligned} \tag{5.5}$$

In the same way we define $\mathscr{R}_2$ by taking into account the coarse space correction

$$\begin{aligned} \mathscr{R}_2 : \prod_{i=0}^{N} \mathbb{R}^{n_i} &\to \mathbb{R}^n \\ (u_i)_{0\le i\le N} &\mapsto \sum_{i=0}^{N} R_i^\top u_i. \end{aligned} \tag{5.6}$$

We note that the subscripts 1 and 2 in $\mathscr{R}_1$ and $\mathscr{R}_2$ refer to one-level and two-level interpolation operators respectively. The following example of two-subdomains-partitioned A illustrates our notation. Let A be given as

$$A = \begin{pmatrix} a_{11} & a_{12} & & \\ a_{21} & a_{22} & a_{23} & \\ & a_{32} & a_{33} & a_{34} \\ & & a_{43} & a_{44} \end{pmatrix}.$$

Then, $\mathcal{N} = \{1, 2, 3, 4\}$. The sets of interior DOF of subdomains are $\mathcal{N}_{1,0} = \{1, 2\}$, $\mathcal{N}_{2,0} = \{3, 4\}$. The sets of overlapping DOF of subdomains are $\Delta_1 = \{3\}$, $\Delta_2 = \{2\}$. The sets of concatenation of the interior DOF and the overlapping DOF of subdomains are $\mathcal{N}_1 = \{1, 2, 3\}$, $\mathcal{N}_2 = \{3, 4, 2\}$. The restriction operator on the interior DOF of subdomains is

$$R_{1,0} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \qquad R_{2,0} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The restriction operator on the overlapping DOF of subdomains is

$$R_{1,\delta} = \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}, \qquad R_{2,\delta} = \begin{pmatrix} 0 & 1 & 0 & 0 \end{pmatrix}.$$

The restriction operator on the concatenation of the interior DOF and the overlapping DOF is

$$R_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \qquad R_2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

The permutation matrix associated with each subdomain is

$$\mathcal{P}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad \mathcal{P}_2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

The permuted matrix associated with each subdomain is

$$\mathcal{P}_1 A \mathcal{P}_1^\top = \begin{pmatrix} a_{11} & a_{12} & & \\ a_{21} & a_{22} & a_{23} & \\ & a_{32} & a_{33} & a_{34} \\ & & a_{43} & a_{44} \end{pmatrix}, \qquad \mathcal{P}_2 A \mathcal{P}_2^\top = \begin{pmatrix} a_{33} & a_{34} & a_{32} & \\ a_{43} & a_{44} & & \\ a_{23} & & a_{22} & a_{21} \\ & & a_{12} & a_{11} \end{pmatrix}.$$

Finally, the algebraic partition of unity can be defined as

$$D_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix}, \qquad D_2 = \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix}.$$

We note that the reordering of lines in the partition of unity matrices $(D_i)_{1 \le i \le N}$ has to be adapted with the lines reordering of $(R_i)_{1 \le i \le N}$ such that (5.4) holds.

5.2 Background

In this section, we start by presenting three lemmas that help compare two symmetric positive definite (or semidefinite) matrices. Then, we review generalities of one- and two-level additive Schwarz preconditioners.

5.2.1 Auxiliary lemmas

The Lemma 4 can be found in [18, Lemma 7.3, p. 164]. This lemma helps prove the effect of the additive Schwarz preconditioner on the largest eigenvalues of the preconditioned operator.

Lemma 4. *Let $A_1, A_2 \in \mathbb{R}^{n \times n}$ be two symmetric positive definite matrices. Suppose that there is a constant $c_u > 0$ such that,*

$$v^\top A_1 v \le c_u \, v^\top A_2 v, \quad \forall v \in \mathbb{R}^n. \tag{5.7}$$

Then the eigenvalues of $A_2^{-1} A_1$ are strictly positive and bounded from above by c_u.

The Lemma 5 is widely known in the community of domain decomposition by the *Fictitious subspace lemma*.

Lemma 5 (Fictitious subspace lemma). *Let $A \in \mathbb{R}^{n_A \times n_A}$, $B \in \mathbb{R}^{n_B \times n_B}$ be two symmetric positive definite matrices. Let $\mathscr{R}$ be an operator defined as*

$$\begin{aligned} \mathscr{R} \; : \; & \mathbb{R}^{n_B} \to \mathbb{R}^{n_A} \\ & v \mapsto \mathscr{R} v, \end{aligned} \tag{5.8}$$

and let $\mathscr{R}^\top$ be its transpose. Suppose that the following conditions hold:

1. *The operator $\mathscr{R}$ is surjective.*
2. *There exists $c_u > 0$ such that*

$$(\mathscr{R} v)^\top A (\mathscr{R} v) \leq c_u \; v^\top B v, \quad \forall v \in \mathbb{R}^{n_B}. \tag{5.9}$$

3. *There exists $c_l > 0$ such that $\forall v_{n_A} \in \mathbb{R}^{n_A}, \exists v_{n_B} \in \mathbb{R}^{n_B} | v_{n_A} = \mathscr{R} v_{n_B}$ and*

$$c_l \; v_{n_B}^\top B v_{n_B} \leq \left(\mathscr{R} v_{n_B}\right)^\top A \left(\mathscr{R} v_{n_B}\right) = v_{n_A}^\top A v_{n_A}. \tag{5.10}$$

Then, the spectrum of the operator $\mathscr{R} B^{-1} \mathscr{R}^\top A$ is contained in the segment $[c_l, c_u]$.

Proof. We refer the reader to [18, Lemma 7.4 p.164] or [52, 53, 28] for a detailed proof. □

We note that there is a general version of Lemma 5 for infinite dimensions. This lemma plays a crucial role in bounding the condition number of our preconditioned operator. The operator $\mathscr{R}$ will stand for the interpolation operator. The matrix B will stand for the block diagonal operator of local subdomain problems. It is important to note that in the finite dimension the existence of the constants c_u and c_l are guaranteed. This is not the case in the infinite dimension spaces. In the finite dimension case, the hard part in the fictitious subspace lemma is to find $\mathscr{R}$ such that c_u/c_l is independent of the number of subdomains. When $\mathscr{R}$ and B are chosen to form the one- or two-level additive Schwarz operator, the first two conditions are satisfied for an upper bound c_u independent of the number of subdomains. An algebraic proof which depends only on the coefficient matrix can be found in [18]. However, the third condition is still an open question if no information from the underlying PDE is used. In this paper we address the problem of defining algebraically a surjective interpolation operator of the two-level additive Schwarz operator such that the third condition holds for a c_l independent of the number of subdomains. This is related to the *stable decomposition property* which was introduced in [40]. Later, in [18], the authors proposed a stable decomposition with the additive Schwarz. This decomposition was based on the underlying PDE. Thus, when only the coefficient matrix A is known, this decomposition is not possible to be computed.

The two following lemmas will be applied to choose the local vectors that contribute to the coarse space. They are based on low rank corrections. In [18], the authors present two lemmas [18, Lemma 7.6 p.167, Lemma 7.7 p.168] similar to the following lemmas.

The rank correction proposed in their version is not of minimal rank. We modify these two lemmas to obtain the smallest rank correction.

Lemma 6. *Let $A, B \in \mathbb{R}^{m\times m}$ be two symmetric positive matrices. Let $ker(A)$, $range(A)$ denote the null space and the range of A respectively. Let $ker(B)$ denote the kernel of B. Let $L = ker(A) \cap ker(B)$, we note $L^{\perp_{ker(A)}}$ the orthogonal complementary of L in $ker(A)$. Let P_0 be an orthogonal projection on $range(A)$. Let τ be a strictly positive real number. Consider the following generalized eigenvalue problem,*

$$\begin{aligned} &Find\ (u_k, \lambda_k) \in range(A) \times \mathbb{R}\ such\ that \\ &P_0 B P_0 u_k = \lambda_k A u_k. \end{aligned} \tag{5.11}$$

Let P_τ be an orthogonal projection on the subspace

$$Z = L^{\perp_{ker(A)}} \oplus span\{u_k \,|\, \lambda_k > \tau\},$$

then, the following inequality holds:

$$(u - P_\tau u)^\top B (u - P_\tau u) \leq \tau u^\top A u,\ \forall u \in \mathbb{R}^m. \tag{5.12}$$

Furthermore, Z is the subspace of smallest dimension such that (5.12) holds.

Proof. Let $m_A = dim(range(A))$. Let

$$\lambda_1 \leq \ldots \leq \lambda_{m_\tau} \leq \tau < \lambda_{m_\tau+1} \leq \ldots \leq \lambda_{m_A}$$

be the eigenvalues of the generalized eigenvalue problem (5.11). Let

$$u_1, \ldots, u_{m_\tau}, u_{m_\tau+1}, \ldots, u_{m_A}$$

be the corresponding eigenvectors, A-orthonormalized. Let $k_B = dim(ker(B) \cap ker(A))$, $k_A = dim(ker(A)) = m - m_A$. Let $v_1, \ldots, v_{k_B}$ be an orthogonal basis of L and let $v_{k_B+1}, \ldots, v_{k_A}$ be an orthogonal basis of $L^{\perp_{ker(A)}}$ such that $v_1, \ldots, v_{k_A}$ is an orthogonal basis of $ker(A)$. The symmetry of A and B permits to have

$$\begin{aligned} u_i^\top A u_j &= \delta_{ij}, \quad 1 \leq i, j \leq m_A, \\ u_i^\top B u_j &= \lambda_i \delta_{ij}, \quad 1 \leq i, j \leq m_A, \\ v_i^\top v_j &= \delta_{ij}, \quad 1 \leq i, j \leq k_A, \\ L &= \text{span}\left\{v_1, \ldots, v_{k_B}\right\}, \\ L^{\perp_{ker(A)}} &= \text{span}\left\{v_{k_B+1}, \ldots, v_{k_A}\right\}, \end{aligned}$$

where δ_{ij} stands for the Kronecker symbol. For a vector $u \in \mathbb{R}^m$ we can write:

$$P_0 u = \sum_{k=1}^{m_A} (u_k^\top A P_0 u) u_k.$$

Then, we have

$$P_\tau u = u - P_0 u - \sum_{k=1}^{k_B} (v_k^\top u) v_k + \sum_{k=m_\tau+1}^{m_A} (u_k^\top A P_0 u) u_k.$$

Thus,

$$u - P_\tau u = \sum_{k=1}^{k_B} (v_k^\top u) v_k + \sum_{k=1}^{m_\tau} (u_k^\top A P_0 u) u_k.$$

Hence, the left side of (5.12) can be written as:

$$\begin{aligned}
(u - P_\tau u)^\top B (u - P_\tau u) &= \left(\sum_{k=1}^{k_B} (v_k^\top u) v_k + \sum_{k=1}^{m_\tau} (u_k^\top A P_0 u) u_k \right)^\top B \left(\sum_{k=1}^{k_B} (v_k^\top u) v_k + \sum_{k=1}^{m_\tau} (u_k^\top A P_0 u) u_k \right), \\
&= \left(\sum_{k=1}^{k_B} (v_k^\top u) v_k + \sum_{k=1}^{m_\tau} (u_k^\top A P_0 u) u_k \right)^\top \left(\sum_{k=1}^{m_\tau} \lambda_k (u_k^\top A P_0 u) A u_k \right), \\
&= \left(\sum_{k=1}^{k_B} (v_k^\top u) A v_k + \sum_{k=1}^{m_\tau} (u_k^\top A P_0 u) A u_k \right)^\top \left(\sum_{k=1}^{m_\tau} \lambda_k (u_k^\top A P_0 u) u_k \right), \\
&= \left(\sum_{k=1}^{m_\tau} (u_k^\top A P_0 u) A u_k \right)^\top \left(\sum_{k=1}^{m_\tau} \lambda_k (u_k^\top A P_0 u) u_k \right), \\
&= \left(\sum_{k|\lambda_k \le \tau} (u_k^\top A P_0 u) u_k \right)^\top \left(\sum_{k|\lambda_k \le \tau} \lambda_k (u_k^\top A P_0 u) A u_k \right), \\
&= \left(\sum_{k|\lambda_k \le \tau} \sum_{j|\lambda_j \le \tau} (u_k^\top A P_0 u) u_k^\top \left(\lambda_j (u_j^\top A P_0 u) A u_j \right) \right), \\
&= \sum_{k|\lambda_k \le \tau} (u_k^\top A P_0 u)^2 \lambda_k.
\end{aligned}$$

We obtain (5.12) by remarking that.

$$\begin{aligned}\sum_{k|\lambda_k\leq\tau}(u_k^\top AP_0u)^2\lambda_k &\leq \tau\sum_{k=1}^{m_A}(u_k^\top AP_0u)^2,\\ &= \tau\sum_{k=1}^{m_A}(u_k^\top AP_0u)(u_k^\top AP_0u),\\ &= \tau(P_0u)^\top AP_0u,\\ &= \tau u^\top Au.\end{aligned}$$

There remains the minimality of the dimension of Z. First, remark that

$$u^\top Bu > \tau u^\top Au, \quad \forall u \in Z.$$

To prove the minimality, suppose that there is a subspace Z_1 of dimension less than the dimension of Z. By this assumption, there is a non-zero vector $w \in (Z \cap Z_1)^{\perp_Z}$, where $(Z \cap Z_1)^{\perp_Z}$ is the orthogonal complementary of $(Z \cap Z_1)$ in Z, such that $w \perp Z_1$. By construction, we have

$$w^\top Bw > \tau w^\top Aw.$$

□

Lemma 7. *Let $A \in \mathbb{R}^{m\times m}$ be a symmetric positive matrix and $B \in \mathbb{R}^{m\times m}$ be an SPD matrix. Let $ker(A)$, $range(A)$ denote the null space and the range of A respectively. Let P_0 be an orthogonal projection on $range(A)$. Let τ be a strictly positive real number. Consider the following generalized eigenvalue problem,*

$$\begin{aligned}&Find\ (u_k,\lambda_k) \in \mathbb{R}^m \times \mathbb{R}\ such\ that\\ &Au_k = \lambda_k Bu_k.\end{aligned} \tag{5.13}$$

Let P_τ be an orthogonal projection on the subspace

$$Z = span\left\{u_k \,|\, \lambda_k < \frac{1}{\tau}\right\},$$

then, the following inequality holds:

$$(u - P_\tau u)^\top B(u - P_\tau u) \leq \tau u^\top Au \ \forall u \in \mathbb{R}^m. \tag{5.14}$$

Z is the subspace of smallest dimension such that (5.14) holds.

Proof. Let $u_1,\ldots,u_{m_0}$ be an orthogonal basis vectors of $ker(A)$. Let

$$0 < \lambda_{m_0+1} \leq \ldots \leq \lambda_{m_\tau} < \frac{1}{\tau} \leq \lambda_{m_\tau+1} \leq \ldots \leq \lambda_m$$

be the eigenvalues strictly larger than 0 of the generalized eigenvalue problem (5.13). Let

$$u_{m_0+1}, \dots, u_{m_\tau}, u_{m_\tau+1}, \dots, u_m$$

be the corresponding eigenvectors A-orthonormalized. We can suppose that

$$\begin{aligned} u_i^\top A u_j &= \delta_{ij}, \quad m_0 + 1 \le i, j \le m, \\ u_i^\top B u_j &= \frac{1}{\lambda_i}\delta_{ij}, \quad m_0 + 1 \le i, j \le m, \\ u_i^\top u_j &= \delta_{ij}, \quad 1 \le i, j \le m_0, \end{aligned}$$

where δ_{ij} stands for the Kronecker symbol. We can write

$$P_0 u = \sum_{k=m_0+1}^{m} (u_k^\top A P_0 u) u_k.$$

Then, we have

$$P_\tau u = u - P_0 u + \sum_{k=m_0+1}^{m_\tau} (u_k^\top A P_0 u) u_k.$$

Thus,

$$\begin{aligned} u - P_\tau u &= \sum_{k=m_\tau+1}^{m} (u_k^\top A P_0 u) u_k, \\ &= \sum_{k|\lambda_k \ge \frac{1}{\tau}} (u_k^\top A P_0 u) u_k. \end{aligned}$$

Hence, the left side of (5.14) can be written

$$\begin{aligned}(u - P_\tau u)^\top B (u - P_\tau u) &= \left(\sum_{k|\lambda_k \geq \frac{1}{\tau}} (u_k^\top A P_0 u) u_k \right)^\top B \left(\sum_{k|\lambda_k \geq \frac{1}{\tau}} (u_k^\top A P_0 u) u_k \right), \\ &= \left(\sum_{k|\lambda_k \geq \frac{1}{\tau}} (u_k^\top A P_0 u) u_k \right)^\top \left(\sum_{k|\lambda_k \geq \frac{1}{\tau}} \frac{1}{\lambda_k} (u_k^\top A P_0 u) A u_k \right), \\ &= \left(\sum_{k|\lambda_k \geq \frac{1}{\tau}} \sum_{j|\lambda_j \geq \frac{1}{\tau}} (u_k^\top A P_0 u) u_k^\top \left(\frac{1}{\lambda_j} (u_j^\top A P_0 u) A u_j \right) \right), \\ &= \sum_{k|\lambda_k \geq \frac{1}{\tau}} (u_k^\top A P_0 u)^2 \frac{1}{\lambda_k}.\end{aligned}$$

We obtain (5.14) by remarking that.

$$\begin{aligned}\sum_{k|\lambda_k \geq \frac{1}{\tau}} (u_k^\top A P_0 u)^2 \frac{1}{\lambda_k} &\leq \tau \sum_{k=1}^{m} (u_k^\top A P_0 u)^2, \\ &= \tau \sum_{k=m_0+1}^{m} (u_k^\top A P_0 u)(u_k^\top A P_0 u), \\ &= \tau (P_0 u)^\top A P_0 u, \\ &= \tau u^\top A u.\end{aligned}$$

There remains the minimality of Z. First, remark that

$$u^\top B u > \tau u^\top A u, \quad \forall u \in Z.$$

To prove the minimality, suppose that there is a subspace Z_1 of dimension less than the dimension of Z. By this assumption, there is a non-zero vector $w \in (Z \cap Z_1)^{\perp_Z}$, where $(Z \cap Z_1)^{\perp_Z}$ is the orthogonal complementary of $(Z \cap Z_1)$ in Z, such that $w \perp Z_1$. By construction, we have

$$w^\top B w > \tau w^\top A w.$$

□

The previous lemmas are general and algebraic and not directly related to the preconditioning. In the following section we will review the one- and two-level additive Schwarz preconditioner.

5.2.2 One- and two-level additive Schwarz preconditioner

In this section we review the definition and general properties of one- and two-level additive Schwarz preconditioners, ASM, ASM_2 respectively. We review, without proving, several lemmas introduced in [13, 18]. These lemmas show how the elements of ASM_2 without any specific property of the coarse space $\mathcal{S}$ verify the conditions 1 and 2 of the fictitious subspace Lemma 5.

The two-level preconditioner ASM_2 with coarse space $\mathcal{S}$ is defined as

$$M_{ASM,2}^{-1} = \sum_{i=0}^{N} R_i^\top \left(R_i A R_i^\top\right)^{-1} R_i. \tag{5.15}$$

If $n_0 = 0$, i.e., the subspace $\mathcal{S}$ is trivial, the term

$$R_0^\top \left(R_0 A R_0^\top\right)^{-1} R_0 = 0$$

by convention.
The following lemma gives the additive Schwarz method a matrix representation as in [18].

Lemma 8. *The additive Schwarz operator can be represented as:*

$$M_{ASM,2}^{-1} = \mathscr{R}_2 \mathcal{B}^{-1} \mathscr{R}_2^\top, \tag{5.16}$$

where $\mathscr{R}_2^\top$ is the operator adjoint of $\mathscr{R}_2$ and $\mathcal{B}$ is a block diagonal operator defined as the following

$$\begin{aligned} \mathcal{B} : \prod_{i=0}^{N} \mathbb{R}^{n_i} &\to \prod_{i=0}^{N} \mathbb{R}^{n_i} \\ (u_i)_{0\le i\le N} &\mapsto \left(\left(R_i A R_i^\top\right) u_i\right)_{0\le i\le N} \end{aligned} \tag{5.17}$$

where $R_i A R_i^\top$ for $0 \le i \le N$ is the i^{th} diagonal block.

Proof. The proof follows directly from the definition of $\mathcal{B}$ and $\mathscr{R}_2$. □

We note that the dimension of the matrix representation of $\mathcal{B}$ is larger than the dimension of A. More precisely, $\mathcal{B}$ has the following dimension

$$n_{\mathcal{B}} = \sum_{i=0}^{N} n_i = n + n_0 + \sum_{i=1}^{N} \delta_i.$$

The one-level additive Schwarz preconditioner can be defined in the same manner. It corresponds to the case where the subspace $\mathcal{S}$ is trivial. The following Lemma 9, [18, Lemma 7.10, p. 173] states that the operator $\mathscr{R}_2$ is surjective without any specific assumption about the coarse space $\mathcal{S}$.

Lemma 9. *The operator $\mathscr{R}_2$ as defined in (5.6) is surjective.*

Proof. The proof follows from the definition of $\mathscr{R}_2$ (5.6) and the definition of the partition of unity (5.4). □

Lemma 9 shows that the interpolation operator $\mathscr{R}_2$ seen as a matrix verifies the condition 1 in Lemma 5. The following Lemma 10 guarantees that the matrix representation of the additive Schwarz verifies condition 2 in Lemma 5.

Lemma 10. *Let k_c be the minimum number of distinct colours so that $\left(span\{R_i^\top\}\right)_{1\leq i\leq N}$ of the same colour are mutually A-orthogonal. Then, we have*

$$(\mathscr{R}_2 u_{\mathcal{B}})^\top A\,(\mathscr{R}_2 u_{\mathcal{B}}) \leq (k_c+1)\sum_{i=0}^{N} u_i^\top \left(R_i A R_i^\top\right) u_i, \quad \forall u_{\mathcal{B}} = (u_i)_{0\leq i\leq N} \in \prod_{i=0}^{N} \mathbb{R}^{n_i}. \tag{5.18}$$

Proof. We refer the reader to [13, Theorem 12 p.93] for a detailed proof. □

We note that Lemma 10 is true for any coarse space $\mathcal{S}$, especially when this subspace is trivial. This makes the lemma applicable also for the one-level additive Schwarz preconditioner (the constant on the right-hand side in Lemma 10 becomes k_c). Lemma 11 is the first step to obtain a reasonable constant c_l that verifies the third condition in Lemma 5

Lemma 11. *Let $u_A \in \mathbb{R}^{n_A}$ and $u_{\mathcal{B}} = (u_i)_{0\leq i\leq N} \in \prod_{i=0}^{N} \mathbb{R}^{n_i}$ such that $u_A = \mathscr{R}_2 u_{\mathcal{B}}$. The additive Schwarz operator without any other restriction on the coarse space $\mathcal{S}$ verifies the following inequality*

$$\sum_{i=0}^{N} u_i^\top \left(R_i A R_i^\top\right) u_i \leq 2\, u_A^\top A u_A + (2k_c+1)\sum_{i=1}^{N} u_i^\top R_i A R_i^\top u_i, \tag{5.19}$$

where k_c is defined in Lemma 10.

Proof. We refer the reader to [18, Lemma 7.12, p. 175] to view the proof in detail. □

In order to apply the fictitious subspace Lemma 5, the term $\sum_{i=1}^{N} u_i^\top R_i A R_i^\top u_i$ in the right-hand side of the (5.19) must be bounded by a factor of $u_A^\top A u_A$. For this aim, the next section presents an algebraic local decomposition of the matrix A. Combining this decomposition with the Lemma 6 or Lemma 7 (depending on the definiteness) defines a class of local generalized eigenvalue problems. By solving them, we can define a coarse space $\mathcal{S}$. The additive Schwarz preconditioner combined with $\mathcal{S}$ satisfy the three conditions of the fictitious subspace Lemma 5. Hence, we can control the condition number of the preconditioned system.

5.3 Algebraic local SPSD splitting of an SPD matrix

In this section we present our main contribution. We introduce the *algebraic local SPSD splitting of an SPD matrix* related to a subdomain. Then, we characterize all the algebraic local SPSD splittings of A that are related to each subdomain. We give a non-trivial bound from below for the energy norm of a vector by a locally determined quantity.

We start by defining the algebraic local SPSD splitting of a matrix related to a subdomain.

Definition 2 (Algebraic local SPSD splitting of A related to a subdomain)**.** *Following the previous notations, let $\tilde{A}_i$ be the matrix defined as*

$$\mathcal{P}_i \tilde{A}_i \mathcal{P}_i^\top = \begin{pmatrix} R_{i,0} A R_{i,0}^\top & R_{i,0} A R_{i,\delta}^\top & \\ R_{i,\delta} A R_{i,0}^\top & \tilde{A}_\delta & \\ & & 0 \end{pmatrix}, \tag{5.20}$$

where $\tilde{A}_\delta \in \mathbb{R}^{\delta_i \times \delta_i}$. We say that $\tilde{A}_i$ is an algebraic local SPSD splitting of A related to the subdomain i if the following condition holds

$$0 \leq u^\top \tilde{A}_i u \leq u^\top A u, \quad \forall u \in \mathbb{R}^n. \tag{5.21}$$

Lemma 12. *Let m_1, m_2, m_3 be integers and $m = m_1 + m_2 + m_3$, let $B \in \mathbb{R}^{m \times m}$ be a 3×3 block tridiagonal SPD matrix*

$$B = \begin{pmatrix} B_{11} & B_{12} & \\ B_{21} & B_{22} & B_{23} \\ & B_{32} & B_{33} \end{pmatrix}, \tag{5.22}$$

where $B_{ii} \in \mathbb{R}^{m_i \times m_i}$ is non-trivial matrix for $i \in \{1, 2, 3\}$. Let $\tilde{B}_1 \in \mathbb{R}^{m \times m}$ be

$$\tilde{B}_1 = \begin{pmatrix} B_{11} & B_{12} & \\ B_{21} & \tilde{B}_{22} & \\ & & 0 \end{pmatrix}, \tag{5.23}$$

where $\tilde{B}_{22} \in \mathbb{R}^{m_2 \times m_2}$ is a symmetric matrix verifying the following inequalities

$$u^\top B_{21} B_{11}^{-1} B_{12} u \leq u^\top \tilde{B}_{22} u \leq u^\top \left(B_{22} - B_{23} B_{33}^{-1} B_{32} \right) u, \quad \forall u \in \mathbb{R}^{m_2}, \tag{5.24}$$

then, the following inequality holds

$$0 \leq u^\top \tilde{B}_1 u \leq u^\top B u, \quad \forall u \in \mathbb{R}^m. \tag{5.25}$$

Proof. Consider the difference matrix $F = B - \tilde{B}_1$. Let $F_2 \in \mathbb{R}^{(m_2+m_3)\times(m_2+m_3)}$ be the lowest 2×2 sub-block diagonal matrix of F, i.e.,

$$F_2 = \begin{pmatrix} B_{22} - \tilde{B}_{22} & B_{23} \\ B_{32} & B_{33} \end{pmatrix}.$$

F_2 admits the following decomposition,

$$F_2 = \begin{pmatrix} I & B_{23}B_{33}^{-1} \\ & I \end{pmatrix} \begin{pmatrix} B_{22} - \tilde{B}_{22} - B_{23}B_{33}^{-1}B_{32} & \\ & B_{33} \end{pmatrix} \begin{pmatrix} I & \\ B_{33}^{-1}B_{32} & I \end{pmatrix}. \tag{5.26}$$

Since $\tilde{B}_{22}$ satisfies, by assumption, the inequality (5.24), F_2 satisfies the following inequality

$$0 \leq u^\top F_2 u \quad \forall u \in \mathbb{R}^{(m_2+m_3)},$$

This proves the right inequality in (5.25).

Let $E \in \mathbb{R}^{(m_1+m_2)\times(m_1+m_2)}$ be the upper 2×2 sub-block diagonal of $\tilde{B}_1$. E admits the following decomposition,

$$E = \begin{pmatrix} I & \\ B_{21}B_{11}^{-1} & I \end{pmatrix} \begin{pmatrix} B_{11} & \\ & \tilde{B}_{22} - B_{21}B_{11}^{-1}B_{12} \end{pmatrix} \begin{pmatrix} I & B_{11}^{-1}B_{12} \\ & I \end{pmatrix}. \tag{5.27}$$

The positivity of $\tilde{B}_1$ follows directly from (5.24). □

Lemma 13. *Using the notations from Lemma 12, the following holds*

- *The condition (5.24) in Lemma 12 is not trivial, i.e., the set of matrices $\tilde{B}_1$ that verify the condition (5.24) is not empty*
- *There exist matrices, $\tilde{B}_{22}$, that verify the condition (5.24) with strict inequalities*
- *The left inequality in condition (5.24) is optimal, i.e., if there exists a non-zero vector $u_2 \in \mathbb{R}^{m_2}$ that verifies*

  $$u_2^\top B_{21}B_{11}^{-1}B_{12}u_2 > u_2^\top \tilde{B}_{22}u_2.$$

  Then, there exists a non-zero vector $u \in \mathbb{R}^m$ such that

  $$u^\top \tilde{B}_1 u < 0$$

- *The right inequality in condition (5.24) is optimal, i.e., if there exists a non-zero vector $u_2 \in \mathbb{R}^{m_2}$ that verifies*

  $$u_2^\top \tilde{B}_{22}u_2 > u_2^\top \left(B_{22} - B_{23}B_{33}^{-1}B_{32}\right)u_2.$$

  Then, there exists a non-zero vector $u \in \mathbb{R}^m$ such that

  $$u^\top \tilde{B}_1 u > u^\top B u$$

Proof. First we prove the non-triviality of the set of matrices verifying (5.24). Indeed, let $S(B_{22})$ be the Schur complement of B_{22} in B, namely

$$S(B_{22}) = B_{22} - B_{21}B_{11}^{-1}B_{12} - B_{23}B_{33}^{-1}B_{32}.$$

Set $\tilde{B}_{22} := \frac{1}{2}S(B_{22}) + B_{21}B_{11}^{-1}B_{12}$. Then we have,

$$\tilde{B}_{22} - B_{21}B_{11}^{-1}B_{12} = \left(B_{22} - B_{23}B_{33}^{-1}B_{32}\right) - \tilde{B}_{22} = \frac{1}{2}S(B_{22}),$$

which is an SPD matrix. Hence, the strict inequalities in (5.24) follow.

Let $u_2 \in \mathbb{R}^{m_2}$ be a vector such that

$$u_2^\top B_{21}B_{11}^{-1}B_{12}u_2 > u_2^\top \tilde{B}_{22}u_2,$$

The block-LDLT factorization (5.27) shows that

$$u^\top \tilde{B}_1 u = u_2^\top \left(\tilde{B}_{22} - B_{21}B_{11}^{-1}B_{12}\right)u_2 < 0,$$

where u is defined as

$$u = \begin{pmatrix} I & B_{11}^{-1}B_{12} \\ & I \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ u_2 \end{pmatrix}.$$

In the same manner we verify the optimality mentioned in the last point. □

Remark 1. *We note that the matrix* $\begin{pmatrix} B_{11} & B_{12} \\ B_{21} & \tilde{B}_{22} \end{pmatrix}$ *defines a seminorm in* $\mathbb{R}^{m_1+m_2}$. *Furthermore, if* $\tilde{B}_{22}$ *is set such that the left inequality in* (5.24) *is strict, then the seminorm becomes a norm.*

For $i \in \{1, \ldots, N\}$, the matrix $\mathcal{P}_i A \mathcal{P}_i^\top$ has the form of a block tridiagonal matrix (the permutation matrix $\mathcal{P}_i$ is defined in the section Notation). The first diagonal block corresponds to the interior DOF of the subdomain i, the second diagonal block corresponds to the overlapping DOF in the subdomain i, and the third block diagonal is associated to the rest of the DOF. This means that we can apply Lemma 12 on each subdomain by considering its interior DOF, overlapping DOF and the rest of the DOF.

Proposition 6. *For each subdomain* $i \in \{1, \ldots, N\}$, *let* $\tilde{A}_i \in \mathbb{R}^{n\times n}$ *be defined as*

$$\mathcal{P}_i \tilde{A}_i \mathcal{P}_i^\top = \begin{pmatrix} R_{i,0}AR_{i,0}^\top & R_{i,0}AR_{i,\delta}^\top & \\ R_{i,\delta}AR_{i,0}^\top & \tilde{A}_\delta^i & \\ & & 0 \end{pmatrix}, \tag{5.28}$$

where $\tilde{A}_\delta^i \in \mathbb{R}^{\delta_i\times\delta_i}$ *satisfies the following conditions*

$\forall u \in \mathbb{R}^{\delta_i}$,

- $u^\top \left(R_{i,\delta}AR_{i,0}^\top\right)\left(R_{i,0}AR_{i,0}^\top\right)^{-1}\left(R_{i,0}AR_{i,\delta}^\top\right)u \le u^\top \tilde{A}_\delta^i u$
- $u^\top \tilde{A}_\delta^i u \le u^\top \left(\left(R_{i,\delta}AR_{i,\delta}^\top\right) - \left(R_{i,\delta}AR_{i,c}^\top\right)\left(R_{i,c}AR_{i,c}^\top\right)^{-1}\left(R_{i,c}AR_{i,\delta}^\top\right)\right)u.$

Then, $\forall i \in \{1, \ldots, N\}$ *the matrix* $\tilde{A}_i$ *is an algebraic local SPSD splitting of* A *related to the*

subdomain i. Moreover, the following inequality holds,

$$0 \leq \sum_{i=1}^{N} u^\top \tilde{A}_i u \leq k_m u^\top A u \quad \forall u \in \mathbb{R}^n, \tag{5.29}$$

where k_m is a number bounded by N.

Proof. Lemma 12 shows that $\tilde{A}_i$ is an algebraic local SPSD splitting of A related to the subdomain i. The inequality (5.29) holds with the constant N for all algebraic local SPSD splittings of A. Thus, depending on the SPSD splitting related to each subdomain there exists a number $k_m \leq N$ such that the inequality holds. □

We note that the matrix $\tilde{A}_i$ is considered local since it has non-zero elements only in the overlapping subdomain i. More precisely,

$$\forall\ j,\ k\ \in \mathcal{N}\ \mid j \notin \mathcal{N}_i\ \vee\ k \notin \mathcal{N}_i,\ \tilde{A}_i(j,k) = 0.$$

Proposition 6 shows that the A-norm of a vector $v \in \mathbb{R}^n$ can be bounded from below by a sum of local seminorms, Remark 1.

5.4 Algebraic stable decomposition with $\mathscr{R}_2$

In the previous section we introduced the algebraic local SPSD splitting of A. In this section we present the *τ-filtering* subspace that is associated with each SPSD splitting. In each subdomain a τ-filtering subspace will contribute to the coarse space. We show how this leads to a class of stable decomposition with $\mathscr{R}_2$. We note that the previous results of 5.2 hold for any coarse space $\mathcal{S}$. Those results are sufficient to determine the constant c_u in the second condition of the fictitious subspace lemma, Lemma 5. However, they do not allow to control the constant c_l of the third condition of the same lemma.

As we will see, the GenEO coarse space [70, 18] corresponds to a special SPSD splitting of A. Therefore, we follow the presentation in [18] in the construction of the coarse space. We note that the proof of Theorem 5 is similar to the proof of [18, Theorem 7.17, p.177]. We present it for the sake of completeness.

Definition 3. *Let $\tilde{A}_i$ be an algebraic local SPSD splitting of A related to the subdomain i, for $i = 1,\ldots,N$. Let $\tau > 0$. Let $\tilde{Z}_i \subset \mathbb{R}^{n_i}$ be a subspace and let $\tilde{P}_i$ be an orthogonal projection on $\tilde{Z}_i$. We say that $\tilde{Z}_i$ is a τ-filtering subspace if*

$$u_i^\top \left(R_i A R_i^\top\right) u_i \leq \tau\ (R_i u)^\top \left(R_i \tilde{A}_i R_i^\top\right)(R_i u), \quad \forall u \in \mathbb{R}^n,$$

where $u_i = \left(D_i \left(I_{n_i} - \tilde{P}_i\right) R_i u\right)$ and D_i is the partition of unity, for $i = 1,\ldots,N$.

After the characterization of the local SPSD splitting of A related to each subdomain, we characterize the associated smallest τ-filtering subspace.

Lemma 14. *Let $\tilde{A}_i$ be an algebraic local SPSD splitting of A related to the subdomain i, for $i = 1,\dots,N$. Let $\tau > 0$. For all subdomains $1 \leq i \leq N$, let*

$$\tilde{G}_i = D_i \left(R_i A R_i^\top \right) D_i,$$

where D_i is the partition of unity. Let $\tilde{P}_{0,i}$ be the projection on $range(R_i \tilde{A}_i R_i^\top)$ parallel to $ker(R_i \tilde{A}_i R_i^\top)$. Let $K = ker(R_i \tilde{A}_i R_i^\top)$, $L = ker(\tilde{G}_i) \cap K$, and $L^{\perp_K}$ the orthogonal complementary of L in K.

- *If $\tilde{G}_i$ is indefinite, consider the following generalized eigenvalue problem*

$$\begin{aligned} &Find\ (u_{i,k},\ \lambda_{i,k}) \in range(R_i \tilde{A}_i R_i^\top) \times \mathbb{R} \\ &such\ that\ \tilde{P}_{0,i} \tilde{G}_i \tilde{P}_{0,i} u_{i,k} = \lambda_{i,k} R_i \tilde{A}_i R_i^\top u_{i,k}. \end{aligned}$$

  Set

$$\tilde{Z}_{\tau,i} = L^{\perp_K} \oplus span\{u_{i,k} \mid \lambda_{i,k} > \tau\}. \tag{5.30}$$

- *If $\tilde{G}_i$ is definite, consider the following generalized eigenvalue problem*

$$\begin{aligned} &Find\ (u_{i,k},\ \lambda_{i,k}) \in \mathbb{R}^{n_i} \times \mathbb{R} \\ &such\ that\ R_i \tilde{A}_i R_i^\top u_{i,k} = \lambda_{i,k} \tilde{G}_i u_{i,k}. \end{aligned}$$

  Set

$$\tilde{Z}_{\tau,i} = span\left\{ u_{i,k} \mid \lambda_{i,k} < \frac{1}{\tau} \right\}. \tag{5.31}$$

Then, $\tilde{Z}_{\tau,i}$ is the smallest dimension τ-filtering subspace and the following inequality holds

$$u_i^\top \left(R_i A R_i^\top \right) u_i \leq \tau\ (R_i u)^\top \left(R_i \tilde{A}_i R_i^\top \right) (R_i u),$$

where $u_i = \left(D_i \left(I_{n_i} - \tilde{P}_{\tau,i} \right) R_i u \right)$, and $\tilde{P}_{\tau,i}$ is the orhtogonal projection on $\tilde{Z}_{\tau,i}$.

Proof. Direct application of Lemma 6 and Lemma 7. □

We will refer to the smallest dimension τ-filtering subspace as $\tilde{Z}_{\tau,i}$ and to the projection on it as $\tilde{P}_{\tau,i}$. Note that for each algebraic local SPSD splitting of A related to a subdomain i, the τ-filtering subspace $\tilde{Z}_{\tau,i}$ defined in Definition 3 changes. Thus, the projection $\tilde{P}_{\tau,i}$ depends on the algebraic local SPSD splitting of A related to the subdomain i.
In the rest of the paper, the notations $\tilde{Z}_{\tau,i}$ and $\tilde{P}_{\tau,i}$ will be used according to the algebraic local SPSD splitting of A that we deal with and following Lemma 14.

Definition 3 leads us to bound the sum in (5.19) by a sum of scalar products associated to algebraic SPSD splittings of A. Therefore, a factor, which depends on the value of τ, of the scalar product associated to A will bound the inequality in (5.19).

Definition 4 (Coarse space based on algebraic local SPSD splitting of A, (ALS))**.** *Let $\tilde{A}_i$ be an algebraic local SPSD splitting of A related to the subdomain i, for $i = 1,\dots,N$. Let $\tilde{Z}_{\tau,i}$ be the subspace associated to $\tilde{A}_i$ as defined in Lemma 14. We define $\mathcal{S}$ the coarse space based on the algebraic local splitting of A related to each subdomain, as the sum of expanded weighted τ-filtering subspaces associated to the algebraic local splitting of A related to each subdomain,*

$$\mathcal{S} = \bigoplus_{i=1}^{N} R_i^\top D_i \tilde{Z}_{\tau,i}. \tag{5.32}$$

Let $\tilde{Z}_0$ be a matrix whose columns form a basis of $\mathcal{S}$. We denote its transpose by $R_0 = \tilde{Z}_0^\top$.

As mentioned previously, the key point to apply the fictitious subspace lemma, Lemma 5, is to find a coarse space that induces a relatively large c_l in the third condition of the lemma. The following theorem proves that ALS satisfies this.

Theorem 5. *Let $\tilde{A}_i$ be an algebraic local SPSD splitting of A related to the subdomain i, for $i = 1,\dots,N$. Let $\tilde{Z}_{\tau,i}$ be the τ-filtering subspace associated to $\tilde{A}_i$, and $\tilde{P}_{\tau,i}$ be the projection on $\tilde{Z}_{\tau,i}$ as defined in Lemma 14. Let $u \in \mathbb{R}^n$ and let $u_i = \left(D_i\left(I_{n_i} - \tilde{P}_{\tau,i}\right)R_i u\right)$ for $i = 1,\dots,N$. Let u_0 be defined as,*

$$u_0 = \left(R_0 R_0^\top\right)^{-1} R_0 \left(\sum_{i=1}^{N} R_i^\top D_i \tilde{P}_{\tau,i} R_i u\right).$$

Let $c_l = (2 + (2k_c + 1)k_m\tau)^{-1}$. Then,

$$u = \sum_{i=0}^{N} R_i^\top u_i,$$

and

$$c_l \sum_{i=0}^{N} u_i^\top R_i A R_i^\top u_i \leq u^\top A u.$$

Proof. Since $\forall y \in \mathcal{S},\ y = R_0^\top \left(R_0 R_0^\top\right)^{-1} R_0 y$, the relation

$$u = \sum_{i=0}^{N} R_i^\top u_i = \mathscr{R}_2\left(u_i\right)_{0\leq i\leq N},$$

follows directly. Lemma 11 shows that

$$\sum_{i=0}^{N} u_i^\top R_i A R_i^\top u_i \leq 2u^\top A u + (2k_c + 1)\sum_{i=1}^{N} u_i^\top \left(R_i A R_i^\top\right) u_i.$$

By using Lemma 14 we can write

$$\sum_{i=0}^{N} u_i^\top R_i A R_i^\top u_i \leq 2u^\top A u + (2k_c+1)\tau \sum_{i=1}^{N} (R_i u)^\top \left(R_i \tilde{A}_i R_i^\top\right)(R_i u).$$

Since $\tilde{A}_i$ is local, we can write

$$\sum_{i=0}^{N} u_i^\top R_i A R_i^\top u_i \leq 2u^\top A u + (2k_c+1)\tau \sum_{i=1}^{N} u^\top \tilde{A}_i u.$$

Then, by applying Proposition 6, we can write

$$\sum_{i=0}^{N} u_i^\top R_i A R_i^\top u_i \leq 2u^\top A u + (2k_c+1)k_m\tau \; u^\top A u,$$

$$\sum_{i=0}^{N} u_i^\top R_i A R_i^\top u_i \leq (2 + (2k_c+1)k_m\tau)\, u^\top A u.$$

□

Theorem 6. *Let M_{ALS} be the two-level ASM preconditioner combined with ALS. The following inequality holds,*

$$\kappa\left(M_{ALS}^{-1}A\right) \leq (k_c+1)\,(2 + (2k_c+1)k_m\tau)$$

Proof. Lemma 9, Lemma 10, and Theorem 5 show that the two-level preconditioner associated with ALS verifies the conditions of the fictitious subspace lemma, Lemma 5. Hence, the eigenvalues of $M_{ALS}^{-1}A$ verify the following inequality,

$$\frac{1}{2 + (2k_c+1)k_m\tau} \leq \lambda\left(M_{ALS}^{-1}A\right) \leq (k_c+1),$$

and the result follows. □

Remark 2. *Since any τ-filtering subspace $\tilde{Z}_i$ can replace $\tilde{Z}_{\tau,i}$ in Theorem 5, the Theorem 6 applies for coarse spaces of the form $\mathcal{S} = \bigoplus_{i=1}^{N} R_i^\top D_i \tilde{Z}_i$. The difference is that the dimension of the coarse space is minimal by choosing $\tilde{Z}_{\tau,i}$, see Lemma 14.*

We note that the previous theorem, Theorem 6, shows that the spectral condition number of the preconditioned system does not depend on the number of subdomains. It depends only on k_c, k_m, and τ. k_c is bounded by the maximum number of neighbors of a subdomain. k_m is a number bounded by the number of subdomains. It depends on the algebraic local SPSD splitting of each subdomain. Partitioned graphs of sparse matrices have structures such that k_c is small. The parameter τ can be chosen small enough such that ALS has a relatively small dimension.

5.4.1 GenEO coarse space

In [18], the authors present the theory of one- and two-level additive Schwarz preconditioners. To bound the largest eigenvalue of the preconditioned system they use the algebraic properties of the additive Schwarz preconditioner. However, to bound the smallest eigenvalue, they benefit from the discretization of the underlying PDE. In the environment of the finite element method, they construct local matrices corresponding to the integral of the operator in the overlapping subdomain. For each subdomain, the expanded matrix has the form

$$\mathcal{P}_i \tilde{A}_i \mathcal{P}_i^\top = \begin{pmatrix} R_{i,0} A R_{i,0}^\top & R_{i,0} A R_{i,\delta}^\top & \\ R_{i,\delta} A R_{i,0}^\top & \tilde{A}_\delta^i & \\ & & 0 \end{pmatrix},$$

where $\tilde{A}_\delta^i$ corresponds to the integral of the operator in the overlapping region with neighbors of the subdomains i. This matrix is SPSD since the global operator is SPD. Since the integral over the subdomain is always smaller than the integral over the global domain (positive integrals), the following inequality holds

$$0 \leq u^\top \tilde{A}_i u \leq u^\top A u, \ \forall u \in \mathbb{R}^n.$$

Hence, Lemma 13 confirms that the matrix $\tilde{A}_i$ corresponds to an algebraic local SPSD splitting of A related to the subdomain i. Thus, GenEO is a member of the class of preconditioners that are based on the algebraic local SPSD splitting of A. We note that the parameter k_m, defined in (5.29), with the algebraic local SPSD splitting of A corresponding to GenEO can be shown to be equal to the maximum number of subdomains sharing a DOF.

5.4.2 Extremum efficient coarse space

In this section we discuss the two obvious choices to have algebraic local SPSD splitting of A. We show how in practice these two choices are costly. However, they have two advantages. The first is that one of these choices gives an answer to the following question that appears in domain decomposition. *How many local vectors must be added to the coarse space in order to bound the spectral condition number by a number defined a priori?* We are able to answer this question in the case where the additive Schwarz preconditioner is to be used. We note that the answer is given without any analytic information. Only the coefficients of the matrix A have to be known. The second advantage is that both choices give an idea of constructing a non-costly algebraic approximation of an ALS.

In the following discussion we disregard the impact of the parameter k_m. Numerical experiments in 5.5 demonstrate that the impact of this parameter can be negligible. We note that this parameter depends only on the algebraic local SPSD splitting and it is bounded by N.

Suppose that we have two SPSD splittings of A related to a subdomain i, $\tilde{A}_i^{(1)}, \tilde{A}_i^{(2)}$,

such that:

$$u^\top \tilde{A}_i^{(1)} u \leq u^\top \tilde{A}_i^{(2)} u, \quad \forall u \in \mathbb{R}^n.$$

We want to compare the number of vectors that contribute to the coarse space for each SPSD splitting. It is clear that a τ-filtering subspace associated to $\tilde{A}_i^{(1)}$ is a τ-filtering subspace associated to $\tilde{A}_i^{(2)}$. Thus, the following inequality holds,

$$dim(\tilde{Z}_{\tau,i}^{(1)}) \geq dim(\tilde{Z}_{\tau,i}^{(1)}),$$

where $\tilde{Z}_{\tau,i}^{(1)}, \tilde{Z}_{\tau,i}^{(2)}$ are the smallest τ-filtering subspaces associated to $\tilde{A}_i^{(1)}, \tilde{A}_i^{(2)}$, respectively. Therefore, Lemma 13 shows that closer we are to the upper bound in (5.24) less vectors will contribute to ALS. Moreover, closer we are to the lower bound in (5.24) more vectors will contribute to ALS. Indeed, the set of algebraic local SPSD splitting of A related to a subdomain i admits a relation of partial ordering.

$$M_1 \leq M_2 \iff u^\top M_1 u \leq u^\top M_2 u, \quad \forall u.$$

This set admits obviously a smallest and a largest element defined by the left and the right bounds in (5.24), respectively.

Hence, the best ALS corresponds to the following algebraic local SPSD splitting of A, for $i = 1, \ldots, N$,

$$\mathcal{P}_i \tilde{A}_i \mathcal{P}_i^\top = \begin{pmatrix} R_{i,0} A R_{i,0}^\top & R_{i,0} A R_{i,\delta}^\top \\ R_{i,\delta} A R_{i,0}^\top & R_{i,\delta} A R_{i,\delta}^\top - \left(R_{i,\delta} A R_{i,c}^\top\right)\left(R_{i,c} A R_{i,c}^\top\right)^{-1}\left(R_{i,c} A R_{i,\delta}^\top\right) \end{pmatrix}. \qquad 0 \tag{5.33}$$

The dimension of the subspace $\tilde{Z}_{\tau,i}$ associated to $\tilde{A}_i$ (5.33) is minimal over all possible algebraic local SPSD splittings of A related to the subdomain i. We remark that this splitting is not a choice in practice since it includes inverting the matrix $\left(R_{i,c} A R_{i,c}^\top\right)$ which is of large size (approximately corresponding to $N-1$ subdomains). We will refer to (5.33) as *the upper bound SPSD splitting*, the associated coarse space will be referred to as *the upper ALS*.

In the same manner, we can find the worst ALS. The corresponding algebraic local SPSD splitting of A related to the subdomain i is the following

$$\mathcal{P}_i \tilde{A}_i \mathcal{P}_i^\top = \begin{pmatrix} R_{i,0} A R_{i,0}^\top & R_{i,0} A R_{i,\delta}^\top \\ R_{i,\delta} A R_{i,0}^\top & \left(R_{i,\delta} A R_{i,0}^\top\right)\left(R_{i,0} A R_{i,0}^\top\right)^{-1}\left(R_{i,0} A R_{i,\delta}^\top\right) \end{pmatrix}. \qquad 0 \tag{5.34}$$

On the contrary of the best splitting (5.33), this splitting is not costly. It includes inverting the matrix $\left(R_{i,0} A R_{i,0}^\top\right)$ which is considered small. However, the dimension of $\tilde{Z}_{\tau,i}$ associated to $\tilde{A}_i$ (5.34) is maximal. It is of dimension δ_i at least. Indeed, a block-LDLT factorization of $R_i \tilde{A}_i R_i^\top$ shows that its null space is of dimension δ_i. We will refer

to (5.34) as *the lower bound SPSD splitting* the associated coarse space will be referred to as *the lower ALS*.

Remark 3. *A convex linear combination of the lower bound and the upper bound of the SPSD splitting is also an algebraic local SPSD splitting.*

$$\alpha \times \textit{the upper bound SPSD splitting} + (1-\alpha) \times \textit{the lower bound SPSD splitting}$$

We refer to it as α-convex SPSD splitting, We refer to the corresponding ALS as the α-convex ALS.

In the following section we propose a strategy to compute an approximation of reasonable ALS that is not costly.

5.4.3 Approximate ALS

As mentioned in 5.4.2, the extremum cases of ALS are not practical choices. Nevertheless, the ALS can be approximated by considering the following strategy. We restrict the matrix $R_{i,c}AR_{i,c}^{\top}$ to the neighbors DOF of the subdomain i through the graph of A such that the restriction of the matrix $R_{i,c}AR_{i,c}^{\top}$ has a dimension $dim_i \leq d \times n_i$, where $d \geq 1$ is a fixed integer. Then we can take a convex linear combination of the lower bound SPSD splitting and the approximation of the upper bound SPSD splitting. For instance, the error bound on this approximation is still an open question. Numerical experiments show that d does not need to be large.

5.5 Numerical experiments

In this section we present numerical experiments for ALS. We denote ASM_{ALS} the two-level additive Schwarz combined with ALS. If it is not specified, the number of vectors deflated by subdomain is fixed to 15. We use the preconditioned CG implemented in MATLAB 2017R to compare the preconditioners. The threshold of convergence is fixed to 10^{-6}. Our test matrices arise from the discretization of two types of challenging problems: linear elasticity and diffusion problems [29, 1, 55]. Our set of matrices are given in Table 5.1. The matrices SKY2D and SKY3D arise from the boundary value problem of the diffusion equation on Ω, the (2-D) unit square and the (3-D) unit cube, respectively:

$$-\text{div}(\kappa(x)\nabla u) = f \qquad \text{in } \Omega, \tag{5.35}$$

$$u = 0 \qquad \text{on } \Gamma_D, \tag{5.36}$$

$$\frac{\partial u}{\partial n} = 0 \qquad \text{on } \Gamma_N. \tag{5.37}$$

They correspond to skyscraper problems. The domain Ω contains several zones of high permeability. These zones are separated from each other. The tensor κ is given by the

Matrix name	Type	n	NnZ	κ
SKY3D	Skyscraper	8000	53000	10^5
SKY2D	Skyscraper	10000	49600	10^6
EL3D	Elasticity	15795	510181	3×10^{11}

Table 5.1: Matrices used for tests. n is the size of the matrix, NnZ is the number of non-zero elements. HPD stands for Hermitian Positive Definite. κ is the condition number related to the second norm.

following relation:

$$\kappa(x) = 10^3([10x_2]+1) \qquad \text{if } [10x_i] \text{ is odd, } i=1,2,$$
$$\kappa(x) = 1 \qquad \text{otherwise.}$$

$\Gamma_D = [0,1]\times\{0,1\}$ in the (2-D) case. $\Gamma_D = [0,1]\times\{0,1\}\times[0,1]$ in the (3-D) case. Γ_N is chosen as $\Gamma_N = \partial\Omega \setminus \Gamma_D$ and n denotes the exterior normal vector to the boundary of Ω. The linear elasticity problem with Dirichlet and Neumann boundary conditions is defined as follows

$$\operatorname{div}(\sigma(u)) + f = 0 \qquad \text{in } \Omega, \tag{5.38}$$
$$u = 0 \qquad \text{on } \Gamma_D, \tag{5.39}$$
$$\sigma(u)\cdot n = 0 \qquad \text{on } \Gamma_N, \tag{5.40}$$

Ω is a unit cube (3-D). The matrix El3D corresponds to this equation discretized using a triangular mesh with $65\times9\times9$ vertices. Γ_D is the Dirichlet boundary, Γ_N is the Neumann boundary, f is a force, u is the unknown displacement field. The Cauchy stress tensor $\sigma(.)$ is given by Hooke's law: it can be expressed in terms of Young's modulus E and Poisson's ration ν. n denotes the exterior normal vector to the boundary of Ω. We consider discontinuous E and ν: $(E_1,\nu_1) = (2\times10^{11}, 0.45)$, $(E_2,\nu_2) = (10^7, 0.25)$. Data elements of this problem are obtained by the application FreeFem++ [34]. Table 5.2 presents a comparison between one-level ASM and ASM_2 with the upper bound ALS. As it is known, the iteration number of CG preconditioned by ASM increases by increasing the number of subdomains. However, we remark that the iteration number of the CG preconditioned by ALS is robust when the number of subdomain increases.

In Table 5.3 we compare three ALS, the upper bound, α_1-convex, and α_2-convex, where $\alpha_1 = 0.75$ and $\alpha_2 = 0.25$. Table 5.3 shows the efficiency of three ALS related to different SPSD splittings.

To illustrate the impact of the parameter k_m, when increasing the number of subdomains, on bounding the spectral condition number, we do the following. We choose τ as

$$\tau = \frac{1}{2}\left(\frac{\tilde{\kappa}}{k_c+1} - 2\right)(2k_c+1)^{-1},$$

i.e., we suppose that k_m has no impact on τ. The resulting spectral condition number will

Matrix	n	N	n_{uC}	n_{ASM}
SKY3D	8000	4	23	29
		8	25	35
		16	25	37
		32	22	55
		64	24	79
		128	24	-
SKY2D	10000	4	18	54
		8	19	-
		16	20	-
		32	22	-
		64	26	-
		128	31	-
EL3D	15795	4	38	-
		8	43	-
		16	51	-
		32	51	-
		64	67	-
		128	92	-

Table 5.2: Comparison between ASM_2 with the upper ALS and one-level additive Schwarz, n is the dimension of the problem, N is the number of subdomains, n_{uC} is the iteration number of CG preconditioned by ASM_2, and n_{ASM} is the iteration number of CG preconditioned by one-level ASM. The sign – means that the method did not converge in fewer than 100 iteration.

Matrix	n	N	n_{uC}	n_{α_1}	n_{α_2}
SKY3D	8000	4	23	22	22
		8	25	25	23
		16	25	24	24
		32	22	22	22
		64	24	23	21
		128	24	24	22
SKY2D	10000	4	18	18	17
		8	19	19	19
		16	20	19	19
		32	22	21	18
		64	26	24	20
		128	31	28	20
EL3D	15795	4	38	38	38
		8	43	43	43
		16	51	51	51
		32	51	51	51
		64	67	67	67
		128	92	92	92

Table 5.3: Comparison between ALS variants, the upper bound ALS, the α_1-convex ALS, and the α_2-convex ALS, n is the dimension of the problem, N is the number of subdomains, the subscript uC refers to the upper bound ALS, $n_{\cdot}$ is the iteration number of ASM_2, α refers to the coefficient in the convex linear combination, $\alpha_1 = 0.75$ and $\alpha_2 = 0.25$.

N	dim_{uC}	n_{uC}	dim_{α_1}	n_{α_1}	dim_{α_2}	n_{α_2}	dim_{Gen}	n_{Gen}
4	82	20	92	19	120	18	106	20
8	179	23	209	20	240	20	229	24
16	304	37	394	30	480	28	391	38
32	447	53	583	45	960	36	614	42
64	622	84	769	73	1920	51	850	55
128	969	131	1096	112	3834	77	1326	61

Table 5.4: Matrix El3D, ALS variants and GenEo coarse space with the minimum number of deflated vectors disregarding the parameter k_m, N is the number of subdomains, the subscript uC refers to the upper bound ALS. $dim_{\cdot}$ is the dimension of ALS, $n_{\cdot}$ is the iteration number of ASM_2, α refers to the coefficient in the convex ALS, $\alpha_1 = 0.75$ and $\alpha_2 = 0.25$, the subscript Gen stands for the GenEO coarse space. See Table 5.5

N	κ_{uC}	κ_{α_1}	κ_{α_2}	κ_{Gen}
4	5	4	4	5
8	8	5	5	7
16	15	10	9	15
32	34	25	15	18
64	100	67	30	31
128	231	178	86	39

Table 5.5: Estimation of the spectral condition number of matrix El3D preconditioned by ASM_2 with ALS variants and GenEo coarse space, results correspond to Table 5.4, N is the number of subdomains, the subscript uC refers to the upper bound ALS, α refers to the coefficient in the convex ALS, $\alpha_1 = 0.75$ and $\alpha_2 = 0.25$, the subscript Gen stands for the GenEO coarse space.

be effected only by the parameter k_m see Table 5.5. Table 5.4 and Table 5.5 present results for ALS variants when $\tilde{\kappa} = 100$. We perform this test on the elasticity problem where we could also compare against the GenEO coarse space [70, 18]. Table 5.4 shows the dimension of ALS of each variant as well as the iteration number for preconditioned CG to reach the convergence tolerance. On the other hand, Table 5.5 shows an estimation of the spectral condition number of the preconditioned system. This estimation is performed by computing an approximation of the largest and the smallest eigenvalues of the preconditioned operator by using the Krylov-Schur method [73] in MATLAB. The same tolerance τ is applied for GenEO. In order to avoid a large-dimension coarse space, 30 vectors at max are deflated per subdomain.

We note that results in Table 5.4 satisfy the discussion in 5.4.2. Indeed the upper bound ALS has the minimum dimension, 0.75- and 0.25-convex ALS follow the upper bound ALS respectively.

Table 5.5 demonstrates the impact of k_m on the bound of the spectral condition number. We notice that its effect increases when α is closer to 1. In Figure 5.1 we present a histogram of the number of deflated vectors by each subdomain. We remark that the

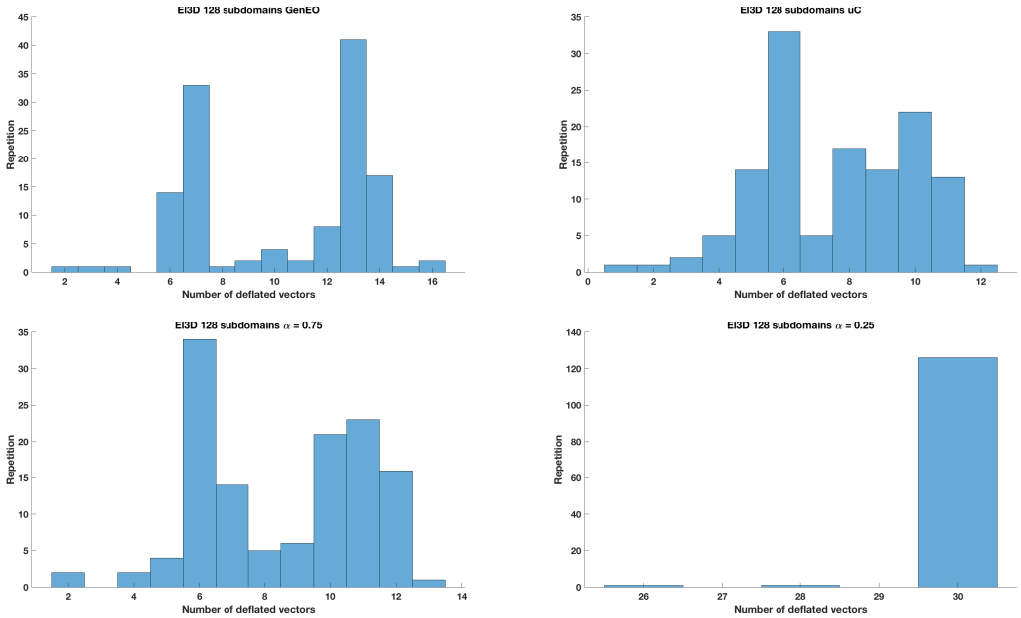

Figure 5.1: Histogram of the number of deflated vectors by each subdomain for different ALS, GenEO; uC, the upper bound ALS; α_1-convex ALS, $\alpha_1 = 0.75$; α_2-convex ALS, $\alpha_2 = 0.25$

number of vectors that each subdomain contributes to the coarse space is not necessarily equal. In the case of α_2-convex ALS, most subdomains reach the maximum number of deflated vectors, 30, that we fixed. Moreover, Figure 5.2 compares the number of deflated vectors in each subdomain for the GenEO subspace and the upper bound ALS. This figure illustrates the relation of partial ordering between the SPSD splitting as discussed in 5.4.2.

In Table 5.6 we show the impact of the approximation strategy that we proposed in 5.4.3. The distance parameter related to the approximation, see 5.4.3, is fixed for each matrix. It is obtained by tuning. The convex linear combination is chosen as $\alpha = 0.01$. Each subdomain contributes 20 vectors to the coarse space. We remark that the approximation strategy gives interesting results with the convection-diffusion problem matrices SKY2D and SKY3D. With a small factor of the local dimension $d = 2$ and $d = 3$, respectively, the approximate ALS is able to perform relatively as efficient as the upper bound ALS. For the elasticity problem with a larger factor $d = 5$, the approximate ALS reduces the iteration number, however, we remark that the latter increases by increasing the number of subdomains.

Matrix	n	N	n_{uC}	d	n_{ap}
SKY3D	8000	4	22	2	22
		8	23		23
		16	24		22
		32	22		22
		64	24		22
		128	22		44
SKY2D	10000	4	17	3	17
		8	18		18
		16	20		19
		32	22		22
		64	26		59
		128	31		90
EL3D	15795	4	27	5	54
		8	36		56
		16	37		77
		32	43		136
		64	61		-
		128	83		-

Table 5.6: Comparison between the upper bound ALS and the approximation strategy presented in 5.4.3, n is the dimension of the problem, N is the number of subdomains, n_{uC} is the iteration number of CG preconditioned by ASM_2 with the upper bound ALS, d stands for the factor of local dimension to approximate the upper bound SPSD splitting, as explained in 5.4.3, and n_{ap} is the iteration number of CG preconditioned by ASM_2 with approximation of ALS, the convex linear combination is chosen as $(0.01 \times \text{approximation of the upper bound} + 0.99 \times \text{lower bound})$. The sign – means that the method did not converge in fewer than 150 iteration.

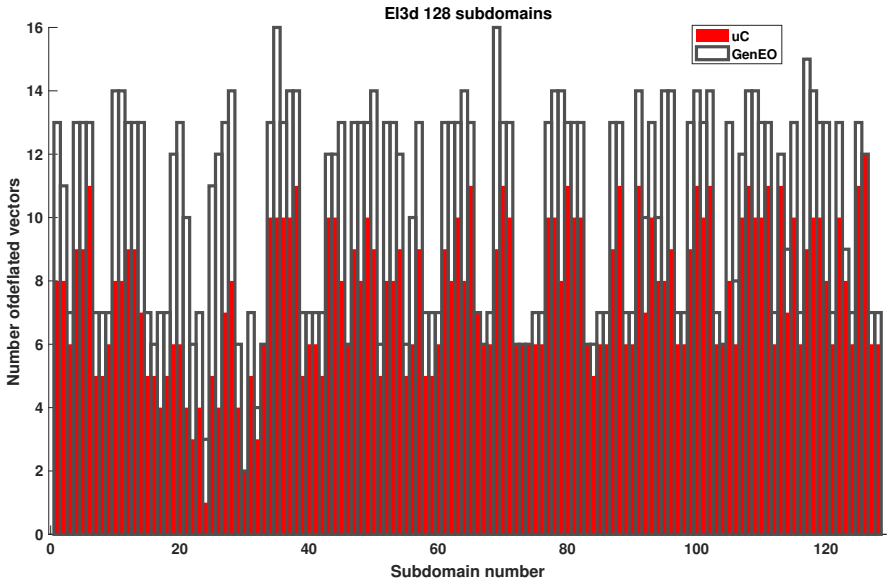

Figure 5.2: Comparaison between the number of deflated vectors per subdomain GenEO coarse space and the upper bound ALS

5.6 Conclusion

In this paper we reviewed generalities of one- and two-level additive Schwarz preconditioner. We introduced the algebraic local SPSD splitting of an SPD matrix A. We characterized all possible algebraic local SPSD splitting. To study the minimality of the dimension of the coarse space, we introduced the τ-filtering subspaces. Based on the algebraic local SPSD splitting and inspired by the GenEO method [70, 18], we introduced a class of algebraic coarse spaces that are constructed locally, ALS. The characterization of algebraic local SPSD splitting of A and the associated τ-filtering subspaces makes an algebraic framework for studying the coarse spaces related to the additive Schwarz method. We proved that the coarse space of GenEO corresponds to a special case of the SPSD splitting. We discussed different types of ALS and suggested a simple method to approximate a valuable coarse space. For matrices issued from the convection-diffusion problem, the simple method that we proposed gave very interesting results.

CONCLUSION

In this thesis we investigated different methods for solving linear systems. Both cases, Symmetric Positive Definite (SPD) and general invertible matrices were taken into account. At first, we proposed a multiple search direction method based on enlarged Krylov subspaces [29], EGMRES. This method adds at maximum t basis vectors per iteration, where t is user defined number. To select the number of added vectors, a detection test based on inexact breakdown is performed. This strategy was introduced previously in the literature [59, 10], however, its practical use was costly. We reformulated the detection test to make it cheaper. We also used a deflation preconditioner that is previously introduced in the literature [20].

Afterwards, we investigated the impact of selecting the deflation subspace on the solution of a sequence of linear systems. The original method, GCRO-DR [57], allows using any deflation subspace. We reviewed two choices of the deflation subspaces based on an approximation of eigenvectors which are proposed in the literature [48]. Then, we proposed a new deflation subspace based on an approximation of singular vectors. We compared the three variants on sequences of linear systems arising from reservoir simulations. The gain related to the proposed subspace, in terms of iteration count, is up to 40%.

The previous methods can be used for general invertible matrices. Concerning the SPD matrices, we reviewed the one- and two-level algebraic additive Schwarz method [13]. We studied the algebraic coarse spaces that allow to upper bound the condition number of the preconditioned matrix by a user defined number. We presented a class of coarse spaces that can be constructed algebraically and without any information from the problem behind the matrix (e.g., the discretization of the PDE). A well-known analytic coarse space, GenEO [70], corresponds to a special case in the presented class of coarse spaces. Furthermore, the presented framework allows to compare the efficiency, in terms of dimension size of the coarse space, of several types of coarse spaces. Numerical experiments and comparisons with GenEO illustrate the theoretical results.

Perspectives

It would be interesting to combine both EGMRES with inexact breakdown detection presented in Chapter 2 with the deflation strategy based on SVD that was investigated

in Chapter 3.

When a large number of subdomains is used in the two-level additive Schwarz preconditioner, even with the best coarse spaces, the dimension of the coarse space would be very large. Factoring the coarse space matrix would not be possible anymore. Using an iterative solver is an attractive candidate in that case. However, this matrix can be very badly conditioned. Thus, it is necessary to precondition this matrix. Based on the theoretical results in Chapter 5 it is possible to add a third level to the additive Schwarz preconditioner. Indeed, the matrix of the coarse space is SPD and has a sparse structure. We would like to investigate in details this third level and the impact of using it on the performance of the preconditioner.

BIBLIOGRAPHY

[1] Y. Achdou and F. Nataf. "Low frequency tangential filtering decomposition". In: *Numerical Linear Algebra with Applications* 14 (2007), pp. 129–147.

[2] E. Agullo, L. Giraud, and Y.-F. Jing. "Block GMRES Method with Inexact Breakdowns and Deflated Restarting". In: *SIAM Journal on Matrix Analysis and Applications* 35.4 (2014), pp. 1625–1651. eprint: http://dx.doi.org/10.1137/140961912.

[3] H. Al Daas and L. Grigori. *A class of efficient locally constructed preconditioners based on coarse spaces*. Research Report RR-9184. Inria – Centre Paris-Rocquencourt ; Laboratoire Jacques-Louis Lions, UPMC, Paris, June 2018.

[4] H. Al Daas, L. Grigori, P. Hénon, and P. Ricoux. *Enlarged GMRES for reducing communication*. Research Report RR-9049. Inria Paris, Mar. 2017.

[5] H. Al Daas, L. Grigori, P. Hénon, and P. Ricoux. "Enlarged GMRES for solving linear systems with one or multiple right-hand sides". In: *IMA Journal of Numerical Analysis* (2018), dry054. eprint: /oup/backfile/content_public/journal/imajna/pap/10.1093_imanum_dry054/5/dry054.pdf.

[6] H. Al Daas, L. Grigori, P. Hénon, and P. Ricoux. *Recycling Krylov subspaces and reducing deflation subspaces for solving sequence of linear systems*. Research Report RR-9206. Inria Paris, Oct. 2018.

[7] W. E. Arnoldi. "The principle of minimized iterations in the solution of the matrix eigenvalue problem". In: *Quarterly of Applied Mathematics* 9.1 (Apr. 1951), pp. 17–29.

[8] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz. "Minimizing Communication in Linear Algebra". In: *CoRR* abs/0905.2485 (2009). arXiv: 0905.2485.

[9] R. Bridson and C. Greif. "A Multipreconditioned Conjugate Gradient Algorithm". In: *SIAM Journal on Matrix Analysis and Applications* 27.4 (2006), pp. 1056–1068. eprint: https://doi.org/10.1137/040620047.

[10] H. Calandra, S. Gratton, R. Lago, X. Vasseur, and L. M. Carvalho. "A Modified Block Flexible GMRES Method with Deflation at Each Iteration for the Solution of Non-Hermitian Linear Systems with Multiple Right-Hand Sides". In: *SIAM Journal on Scientific Computing* 35.5 (2013), S345–S367. eprint: http://dx.doi.org/10.1137/120883037.

[11] H. Cao. "DEVELOPMENT OF TECHNIQUES FOR GENERAL PURPOSE SIMULATORS". PhD thesis. Stanford University, 2002.

[12] T. F. Chan. "Rank revealing QR factorizations". In: *Linear Algebra and its Applications* 88 (1987), pp. 67–82.

[13] T. F. Chan and T. P. Mathew. "Domain decomposition algorithms". In: *Acta Numerica* 3 (1994), pp. 61–143.

[14] M. A. Christie and M. J. Blunt. "Tenth SPE Comparative Solution Project: A Comparison of Upscaling Techniques". In: (2001).

[15] A. Chronopoulos and C. Gear. "s-step iterative methods for symmetric linear systems". In: *Journal of Computational and Applied Mathematics* 25.2 (1989), pp. 153–168.

[16] N. R. Council. *Getting Up to Speed: The Future of Supercomputing*. Ed. by S. L. Graham, M. Snir, and C. A. Patterson. Washington, DC: The National Academies Press, 2005.

[17] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou. "Communication-optimal Parallel and Sequential QR and LU Factorizations". In: *SIAM Journal on Scientific Computing* 34.1 (2012), A206–A239. eprint: https://doi.org/10.1137/080731992.

[18] V. Dolean, P. Jolivet, and F. Nataf. *An introduction to domain decomposition methods*. Algorithms, theory, and parallel implementation. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2015.

[19] J. Dongarra, V. Eijkhout, and A. Kalhan. *Reverse Communication Interface for Linear Algebra Templates for Iterative Methods*. UT-CS-95-291, 1995.

[20] J. Erhel, K. Burrage, and B. Pohl. "Restarted GMRES preconditioned by deflation". In: *Journal of Computational and Applied Mathematics* 69.2 (1996), pp. 303–318.

[21] V. Faber and T. Manteuffel. "Necessary and Sufficient Conditions for the Existence of a Conjugate Gradient Method". In: *SIAM Journal on Numerical Analysis* 21.2 (1984), pp. 352–362. eprint: https://doi.org/10.1137/0721026.

[22] R. W. Freund and M. Malhotra. "A block QMR algorithm for non-Hermitian linear systems with multiple right-hand sides". In: *Linear Algebra and its Applications* 254.1 (1997), pp. 119–157.

[23] P. Ghysels, T. Ashby, K. Meerbergen, and W. Vanroose. "Hiding Global Communication Latency in the GMRES Algorithm on Massively Parallel Machines". In: *SIAM Journal on Scientific Computing* 35.1 (2013), pp. C48–C71. eprint: https://doi.org/10.1137/12086563X.

[24] P. Ghysels and W. Vanroose. "Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm". In: *Parallel Computing* 40.7 (2014). 7th Workshop on Parallel Matrix Algorithms and Applications, pp. 224–238.

[25] L. Giraud, J. Langou, and M. Rozloznik. "The loss of orthogonality in the Gram-Schmidt orthogonalization process". In: *Computers & Mathematics with Applications* 50.7 (2005). Numerical Methods and Computational Mechanics, pp. 1069–1075.

[26] G. H. Golub and C. F. Van Loan. *Matrix Computations (3rd Ed.)* Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[27] C. Greif, T. Rees, and D. B. Szyld. "MPGMRES: a generalized minimum residual method with multiple preconditioners." In: *SIAM Journal on Scientific Computing*. 14 (1993), pp. 461–469.

[28] M. Griebel and P. Oswald. "On the abstract theory of additive and multiplicativeSchwarz algorithms". In: *Numerische Mathematik* 70.2 (Mar. 1995), pp. 163–180.

[29] L. Grigori, S. Moufawad, and F. Nataf. "Enlarged Krylov Subspace Conjugate Gradient Methods for Reducing Communication". In: *SIAM Journal on Matrix Analysis and Applications* 37.2 (2016), pp. 744–773. eprint: https://doi.org/10.1137/140989492.

[30] L. Grigori, F. Nataf, and S. Yousef. *Robust algebraic Schur complement preconditioners based on low rank corrections*. Research Report RR-8557. INRIA, July 2014, p. 18.

[31] T. Gu, X. Liu, Z. Mo, and X. Chi. "Multiple search direction conjugate gradient method I: methods and their propositions". In: *International Journal of Computer Mathematics* 81.9 (2004), pp. 1133–1143. eprint: https://doi.org/10.1080/00207160410001712305.

[32] A. el Guennouni, K. Jbilou, and H. Sadok. "A block version of BiCGSTAB for linear systems with multiple right-hand sides." eng. In: *ETNA. Electronic Transactions on Numerical Analysis [electronic only]* 16 (2003), pp. 129–142.

[33] M. H. Gutknecht. "Block Krylov space methods for linear systems with multiple right-hand sides: an introduction." In: *in: Modern Mathematical Models, Methods and Algorithms for Real World Systems (A.H. Siddiqi, I.S. Duff, and O. Christensen, eds.)* (2007), pp. 420–447.

[34] F. Hecht. "New development in FreeFem++". In: *J. Numer. Math.* 20.3-4 (2012), pp. 251–265.

[35] M. R. Hestenes and E. Stiefel. "Methods of conjugate gradients for solving linear systems." In: *Journal of research of the National Bureau of Standards.* 49 (1952), pp. 409–436.

[36] M. Hoemmen. "Communication-Avoiding Krylov Subspace Methods." PhD thesis. EECS Department, University of California, Berkeley, 2010.

[37] G. Ifrah. *The Universal History of Computing: From the Abacus to Quantum Computing*. New York, NY, USA: John Wiley & Sons, Inc., 2000.

[38] Intel(R). *Math Kernel Library*.

[39] D. Irony, S. Toledo, and A. Tiskin. "Communication lower bounds for distributed-memory matrix multiplication". In: *Journal of Parallel and Distributed Computing* 64.9 (2004), pp. 1017–1026.

[40] X. J. "Theory of Multilevel Methods". PhD thesis. Cornell University, 1989.

[41] H. Jia-Wei and H. T. Kung. "I/O Complexity: The Red-blue Pebble Game". In: *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*. STOC '81. Milwaukee, Wisconsin, USA: ACM, 1981, pp. 326–333.

[42] G. Karypis and V. Kumar. "Multilevelk-way Partitioning Scheme for Irregular Graphs". In: *Journal of Parallel and Distributed Computing* 48.1 (1998), pp. 96–129.

[43] J. Langou. "Iterative methods for solving linear systems with multiple right-hand sides". PhD thesis. CERFACS, 2003.

[44] R. Li and Y. Saad. "Low-Rank Correction Methods for Algebraic Domain Decomposition Preconditioners". In: *SIAM Journal on Matrix Analysis and Applications* 38.3 (2017), pp. 807–828. eprint: https://doi.org/10.1137/16M110486X.

[45] T. Y. Li and Z. Zeng. "A Rank-Revealing Method with Updating, Downdating, and Applications". In: *SIAM Journal on Matrix Analysis and Applications* 26.4 (2005), pp. 918–946. eprint: https://doi.org/10.1137/S0895479803435282.

[46] T. A. M. "On Computable Numbers, with an Application to the Entscheidungsproblem". In: *Proceedings of the London Mathematical Society* s2-42.1 (), pp. 230–265. eprint: https://londmathsoc.onlinelibrary.wiley.com/doi/pdf/10.1112/plms/s2-42.1.230.

[47] R. B. Morgan. "A Restarted GMRES Method Augmented with Eigenvectors". In: *SIAM Journal on Matrix Analysis and Applications* 16.4 (1995), pp. 1154–1171. eprint: http://dx.doi.org/10.1137/S0895479893253975.

[48] R. B. Morgan. "GMRES with Deflated Restarting". In: *SIAM Journal on Scientific Computing* 24.1 (2002), pp. 20–37. eprint: http://dx.doi.org/10.1137/S1064827599364659.

[49] R. B. Morgan. "Restarted block-GMRES with Deflation of Eigenvalues". In: *Appl. Numer. Math.* 54.2 (July 2005), pp. 222–236.

[50] F. Nataf, F. Hecht, P. Jolivet, and C. Prud'Homme. "Scalable Domain Decomposition Preconditioners For Heterogeneous Elliptic Problems". In: *SC13, (Denver, Colorado, United States)* (2013).

[51] F. Nataf, H. Xiang, V. Dolean, and N. Spillane. "A Coarse Space Construction Based on Local Dirichlet-to-Neumann Maps". In: *SIAM Journal on Scientific Computing* 33.4 (2011), pp. 1623–1642. eprint: https://doi.org/10.1137/100796376.

[52] S. V. Nepomnyaschikh. *Decomposition And Fictitious Domains Methods For Elliptic Boundary Value Problems*. 1992.

[53] S. V. Nepomnyaschikh. "Mesh theorems of traces, normalizations of function traces and their inversions". In: *Sov. J. Numer. Anal. Math. Modeling* 6 (1991), pp. 1–25.

[54] A. A. Nikishin and A. Y. Yeremin. "Variable Block CG Algorithms for Solving Large Sparse Symmetric Positive Definite Linear Systems on Parallel Computers, I: General Iterative Scheme". In: *SIAM Journal on Matrix Analysis and Applications* 16.4 (1995), pp. 1135–1153. eprint: http://dx.doi.org/10.1137/S0895479893247679.

[55] Q. Niu, L. Grigori, P. Kumar, and F. Nataf. "Modified tangential frequency filtering decomposition and its fourier analysis". In: *Numerische Mathematik* 116.1 (2010), pp. 123–148.

[56] D. P. O'Leary. "The block conjugate gradient algorithm and related methods". In: *Linear Algebra and its Applications* 29 (1980), pp. 293–322.

[57] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, and S. Maiti. "Recycling Krylov Subspaces for Sequences of Linear Systems". In: *SIAM Journal on Scientific Computing* 28.5 (2006), pp. 1651–1674. eprint: https://doi.org/10.1137/040607277.

[58] M. L. Parks. *THE ITERATIVE SOLUTION OF A SEQUENCE OF LINEAR SYSTEMS ARISING FROM NONLINEAR FINITE ELEMENT ANALYSIS*. 2005.

[59] M. Robbé and M. Sadkane. "Exact and inexact breakdowns in the block GMRES method". In: *Linear Algebra and its Applications* 419.1 (2006), pp. 265–285.

[60] Y. Saad. *Iterative Methods for Sparse Linear Systems*. 2nd. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2003.

[61] Y. Saad and M. H. Schultz. "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems". In: *SIAM Journal on Scientific and Statistical Computing* 7.3 (1986), pp. 856–869. eprint: http://dx.doi.org/10.1137/0907058.

[62] Y. Saad. "A flexible inner-outer preconditioned GMRES algorithm." In: *SIAM Journal on Scientific Computing*. 14 (1993), pp. 461–469.

[63] H. C. S. Schlumberger, H. T. S. S. U., J. W. W. C. Inc., and H. Y. S. Chevron. "Parallel Scalable Unstructured CPR-Type Linear Solver for Resevoir Simulation". In: *SPE 96809* (2005).

[64] R. Schreiber and B. Parlett. "Block Reflectors: Theory and Computation". In: *SIAM Journal on Numerical Analysis* 25.1 (1988), pp. 189–205. eprint: https://doi.org/10.1137/0725014.

[65] V. Simoncini. "On the Convergence of Restarted Krylov Subspace Methods". In: *SIAM Journal on Matrix Analysis and Applications* 22.2 (2000), pp. 430–452. eprint: https://doi.org/10.1137/S0895479898348507.

[66] V. Simoncini and E. Gallopoulos. "Convergence properties of block GMRES and matrix polynomials". In: *Linear Algebra and its Applications* 247 (1996), pp. 97–119.

[67] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra. *MPI-The Complete Reference, Volume 1: The MPI Core*. 2nd. (Revised). Cambridge, MA, USA: MIT Press, 1998.

[68] P. Sonneveld. "CGS, A Fast Lanczos-Type Solver for Nonsymmetric Linear systems". In: *SIAM Journal on Scientific and Statistical Computing* 10.1 (1989), pp. 36–52. eprint: http://dx.doi.org/10.1137/0910004.

[69] N. Spillane. "An Adaptive MultiPreconditioned Conjugate Gradient Algorithm". In: *SIAM Journal on Scientific Computing* 38.3 (2016), A1896–A1918. eprint: https://doi.org/10.1137/15M1028534.

[70] N. Spillane, V. Dolean, P. Hauret, F. Nataf, C. Pechstein, and R. Scheichl. "Abstract robust coarse spaces for systems of PDEs via generalized eigenproblems in the overlaps". In: *Numerische Mathematik* 126.4 (2014), pp. 741–770.

[71] N. Spillane and D. Rixen. "Automatic spectral coarse spaces for robust finite element tearing and interconnecting and balanced domain decomposition algorithms". In: *International Journal for Numerical Methods in Engineering* 95.11 (), pp. 953–990. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.4534.

[72] A. Stathopoulos and K. Wu. "A Block Orthogonalization Procedure with Constant Synchronization Requirements". In: *SIAM Journal on Scientific Computing* 23.6 (2002), pp. 2165–2182. eprint: https://doi.org/10.1137/S1064827500370883.

[73] G. W. Stewart. "A Krylov–Schur Algorithm for Large Eigenproblems". In: *SIAM Journal on Matrix Analysis and Applications* 23.3 (2002), pp. 601–614. eprint: https://doi.org/10.1137/S0895479800371529.

[74] G. W. Stewart. "Updating a Rank-Revealing ULV Decomposition". In: *SIAM Journal on Matrix Analysis and Applications* 14.2 (1993), pp. 494–499. eprint: https://doi.org/10.1137/0614034.

[75] F. Sullivan and J. Dongarra. "Guest Editors' Introduction: The Top 10 Algorithms". In: *Computing in Science & Engineering* 2 (Jan. 2000), pp. 22–23.

[76] J. M. Tang, R. Nabben, C. Vuik, and Y. A. Erlangga. "Comparison of Two-Level Preconditioners Derived from Deflation, Domain Decomposition and Multigrid Methods". In: *Journal of Scientific Computing* 39.3 (2009), pp. 340–370.

[77] A. Toselli and O. Widlund. *Domain Decomposition Methods - Algorithms and Theory*. Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 2005.

[78] B. Vital. "Étude de quelques méthodes de résolution de problèmes lineaires de grande taille sur multiprocesseur". PhD thesis. Rennes 1, 1990.

[79] H. A. van der Vorst. “Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems”. In: *SIAM Journal on Scientific and Statistical Computing* 13.2 (1992), pp. 631–644. eprint: http://dx.doi.org/10.1137/0913035.

[80] J. Wallis, R. Kendall, and T. Little. “Constrained Residual Acceleration of Conjugate Residual Methods”. In: *SPE 13563* (1985).

Solving linear systems arising from reservoirs modelling

Abstract

This thesis presents a work on iterative methods for solving linear systems that aim at reducing the communication in parallel computing. The main type of linear systems in which we are interested arises from a real-life reservoir simulation. Both schemes, implicit and explicit, of modelling the system are taken into account. Three approaches are studied separately. We consider non-symmetric (resp. symmetric) linear systems. This corresponds to the explicit (resp. implicit) formulation of the model problem. We start by presenting an approach that adds multiple search directions per iteration rather than one as in the classic iterative methods. Then, we discuss different strategies of recycling search subspaces. These strategies reduce the global iteration count of a considerable factor during a sequence of linear systems. We review different existing strategies and present a new one. We discuss the parallel implementation of these methods using a low-level language. Numerical experiments for both sequential and parallel implementations are presented. We also consider the algebraic domain decomposition approach. In an algebraic framework, we study the two-level additive Schwarz preconditioner. We provide the algebraic explicit form of a class of local coarse spaces that bounds the spectral condition number of the preconditioned matrix by a number pre-defined.

Keywords: krylov, block methods, inexact breakdown, deflation, recycling, domain decomposition

---

Résolution de systèmes linéaires issus de la modélisation des réservoirs

Résumé

Cette thèse présente un travail sur les méthodes itératives pour résoudre des systèmes linéaires en réduisant les communications pendant les calculs parallèles. Principalement, on est intéressé par les systèmes linéaires qui proviennent des simulations de réservoirs. Trois approches, que l'on peut considérer comme indépendantes, sont présentées. Nous considérons les systèmes linéaires non-symétriques (resp. symétriques), cela correspond au schéma explicite (resp. implicite) du problème modèle. On commence par présenter une approche qui ajoute plusieurs directions de recherche à chaque itération au lieu d'une seule direction comme dans le cas des méthodes classiques. Ensuite, on considère les stratégies de recyclage des espaces de recherche. Ces stratégies réduisent, par un facteur considérable, le nombre d'itérations global pour résoudre une séquence de systèmes linéaires. On fait un rappel des stratégies existantes et l'on en présente une nouvelle. On introduit et détaille l'implémentation parallèle de ces méthodes en utilisant un langage bas niveau. On présente des résultats numériques séquentiels et parallèles. Finalement, on considère la méthode de décomposition de domaine algébrique. Dans un environnement algébrique, on étudie le préconditionneur de Schwarz additif à deux niveaux. On fournit la forme algébrique explicite d'une classe d'espaces grossiers locaux qui bornent le conditionnement par un nombre donné *a priori*.

Mots clés : krylov, méthodes par bloc, inexact breakdown, déflation, recyclage, décomposition de domaine

---

Inria
2 rue Simone Iff – 75012 Paris – France