Fast solution of sparse linear systems with adaptive choice of preconditioners

Zakariae Jorti



Sorbonne Université

École doctorale de Sciences Mathématiques de Paris Centre

Laboratoire Jacques-Louis Lions

Résolution rapide de systèmes linéaires creux avec choix adaptatif de préconditionneurs

Par Zakariae Jorti

Thèse de doctorat de Mathématiques appliquées

Dirigée par Laura Grigori
Co-supervisée par Ani Anciaux-Sedrakian et Soleiman Yousef

Présentée et soutenue publiquement le 03/10/2019

Devant un jury composé de:

M. **Tromeur-Dervout Damien**, Professeur, Université de Lyon, Rapporteur
M. **Schenk Olaf**, Professeur, Università della Svizzera italiana, Rapporteur
M. **Hecht Frédéric**, Professeur, Sorbonne Université, Président du jury
Mme. **Emad Nahid**, Professeur, Université Paris-Saclay, Examinateur
M. **Vasseur Xavier**, Ingénieur de recherche, ISAE-SUPAERO, Examinateur
Mme. **Grigori Laura**, Directrice de recherche, Inria Paris, Directrice de thèse
Mme. **Anciaux-Sedrakian Ani**, Ingénieur de recherche, IFPEN, Co-encadrante de thèse
M. **Yousef Soleiman**, Ingénieur de recherche, IFPEN, Co-Encadrant de thèse

Doctoral School Sciences Mathématiques de Paris Centre
University Department Laboratoire Jacques-Louis Lions
Thesis defended by Zakariae Jorti
In order to become Doctor from Sorbonne Université
Academic Field Applied Mathematics

Fast solution of sparse linear systems with adaptive choice of preconditioners

---

Under the supervision of:

Doctoral advisor:

Dr. Laura Grigori – INRIA, EPI Alpines, LJLL

IFPEN supervisor:

Dr. Ani Anciaux-Sedrakian – IFPEN, Digital Sciences & Technologies Division

IFPEN supervisor:

Dr. Soleiman Yousef – IFPEN, Digital Sciences & Technologies Division

Contents

List of Figures

List of Tables

List of Algorithms

Introduction

General context and scope of application

With the rapid evolution of computing technology during the last thirty years, scientific and engineering computing has become increasing important in many application fields. Such areas include several applications, like geoscience, internal combustion engine, waste water treatment, catalytic fluidized-bed and solid particle solar receiver.

In this kind of scientific applications, solving partial differential equations usually leads to solving sparse linear systems whose size varies according to the cases studied. The solution of these systems generally constitutes the simulation's most consuming stage in terms of memory occupation and computing time and can even contribute up to about 80% of simulation times. In such applications, the resulting linear systems are very large, sparse and ill-conditioned because of anisotropy and data heterogeneity and, generally, iterative solvers are employed.

With iterative methods, an approximate solution is computed by taking into account the tolerance of the convergence chosen by the user. The system-solving is stopped when the tolerance threshold is reached. The numerical efficiency of these methods is strongly related to that of the preconditioner used which constitutes a major part of the solve cost. In fact, efficient preconditioners are necessary to improve the conditioning of linear systems. In this case, the iterative methods used for solving such systems will converge rapidly and in a reasonable time. That being said, there is not only one robust preconditioner that yields best results in all circumstances, and for all application cases. Moreover, many other factors should be taken into consideration in order to assess the efficiency of a preconditioner $\mathbf{M}$ like precision, complexity, the computing time of the preconditioner itself and finally its scalability or how it behaves on very large matrices.

One could say that finding a suitable preconditioner for solving a sparse linear system is "a combination of art and science" *cf.* [123]. Moreover, it is to be emphasized that designing high quality preconditioners is not an easy matter, especially in the case of parallel processing *cf.* [50]. As

mentioned before, iterative methods compute successive approximations to the solution of a linear system. The cost of these algorithms depends on the number of iterations required to reach the stopping criteria defined by the user and the algorithmic cost of each iteration. It can be reduced by using the appropriate solver options and by the choice of a good preconditioner. However, the efficiency of the preconditioner relies on the problem size, heterogeneity, discontinuities of the problem and is difficult to fix them a priori for the whole simulation. Our addressed problems lead to solve complex systems, for which the difficulty level varies during the simulation itself. But nevertheless, the preconditioner and the convergence tolerance choice is set once by the user for the whole simulation. For this reason, we are studying adaptive strategies that will enable either to switch from one preconditioner to another depending on the systems to be solved during a simulation so as to get a preconditioner that is most adequate to the matrix treated at every time step of the simulation, or to build a custom-made preconditioner that consists of several blocks of local preconditioners computed appropriately for principal submatrices of the global matrix such that the treatment of the subdomains may not be the same everywhere. In other terms, certain blocks may require a preconditioner that is more robust than others.

In spite of the existing means for computing the condition number of a matrix (LAPACK library for example provides routines for estimating the condition number of a matrix), this operation remains costly in terms of computing time, mainly for large systems. This is one of the reasons for which we have tried to explore other alternatives, such as error estimators, and to find a link between the algebraic error estimators and the difficulty of solving the system of linear equations. Hence we resort to error estimators as indicators to get the information needed about the systems to be solved, which will help us to choose or even construct the adequate preconditioner of the matrix.

Since estimators help to identify the areas of the domain with significant errors, we propose to adjust the preconditioning according to the sources of error. The first chapter provides a study of such an approach on an experimental basis. The next chapters show, in theory, the types of preconditioners and solvers that can adapt to the complexity of linear systems using information from a posteriori error estimates. This thesis is structured as follows.

In Chapter 1, we first introduce the global algorithm of adaptive choice of preconditioners based on a posteriori error estimators for dynamic simulations in geosciences. The proposed algorithm aims to switch from one preconditioner to another depending on each of the systems to be solved during a simulation so as to get a preconditioner that is most adequate to the matrix treated at each time step of the simulation. In fact, this algorithm establishes choice criteria between two categories of preconditioners depending on the global value of a posteriori error estimators. We give details about the concept, the motivations and the results obtained with the solvers BiCGStab and GMRES, and some preconditioners like Jacobi, ILU(0), CPR. Second, we experimentally investigate the link between the condition number of matrices and the algebraic estimators. Then, we present the local algorithm of adaptive choice of preconditioner based on a posteriori error estimators for dynamic simulations in geosciences. This algorithm uses the domain decomposition concept by partitioning the initial domain into several subdomains according to the values of error estimators on every node of the domain, and then constructs local preconditioners on each subdomain. Here also we will give details about the concept, the motivations and the results obtained with GMRES solver and LU and ILU(0) preconditioners.

In Chapter 2, we discuss a new adaptive approach for iterative solution of sparse linear systems

arising from partial differential equations (PDE) with self-adjoint operators. The idea is to use the a posteriori estimated local distribution of the algebraic error in order to steer and guide the solve process in such way that the algebraic error is reduced more efficiently in the consecutive iterations. We first explain the motivation behind the proposed procedure and show that it can be equivalently formulated as constructing a special combination of preconditioner and initial guess for the original system. We present some numerical experiments in order to identify when the adaptive procedure can be of practical use.

Chapter 3 focuses on alternatives to the adaptive error-based preconditioners, that replace exact inverses by approximate ones for cases where the algebraic error is scattered all over the domain. The variant considered in this chapter employs a robust preconditioning technique called LORASC that bounds the condition number from above. to avoid exactly inverting some large blocks of the matrix. We also derive upper bounds for the condition number of the preconditioned operator with the resulting approximate adaptive preconditioner. Then the preconditioning costs incurred by both exact and approximate adaptive preconditioners are addressed as well. Finally, we assess the feasibility and the reliability of this alternative by some numerical experiments.

In Chapter 4, we introduce a second variant for the adaptive preconditioner. We prove that within a fixed-point iteration scheme, the growth rate of a dominant part of the algebraic error can be controlled with such a preconditioner. More precisely, after deriving an error-based block partitioning of the matrix, where we denote by L the node indices where the algebraic error is large, we bound this dominant part of the error by a seminorm of the error and then demonstrate that the decrease of this latter quantity from an iteration to the other depends on the largest eigenvalue of the L-block of the preconditioned operator $\mathbf{M}^{-1}\mathbf{A}$. We present some numerical results to show that link, and lastly, we test this preconditioner with a PCG solver and compare against the first variant of the adaptive preconditioner.

In Chapter 5, we apply the adaptive approach of Chapter 2 for iterative solution of linear systems arising from some industrial simulations. The numerical framework is different from the previous one, as we focus on linear systems stemming from cell-centered finite volume discretization of single phase flow models. We adapt the starting assumption and consequently the solve procedure to the finite volume method. In addition, the approach presented in this chapter allows using any iterative algebraic solver and is not limited to PCG. Numerical results of a reservoir simulation example for heterogeneous porous media in two dimensions are discussed later in this chapter.

State-of-the-art

PDE Solving

In various mathematical and physical models, we are brought to solve complex non-linear Partial Differential Equations (PDE). These PDEs lead generally to a non-linear system that is solved by an iterative Newton cf. Kelley [86]. At each Newton step, the system is linearized and the generated system is solved by a preconditioned Krylov solver. This phase, which searchs for all the unknowns concurrently, is the one that consumes most of the time and the global solution time depends on the number of iterations necessary for the convergence. More specifically, at a certain time step t^{n+1}, we want to solve the problem

$$\mathcal{F}(X^{n+1}) = 0 \tag{1}$$

where $\mathcal{F}$ is a nonlinear operator. For this purpose, we are trying to find an approximate solution X^{n+1} by linearizing starting from the solution X^n obtained during the previous time step t^n:

$$\mathcal{F}(X^{n+1}) = J_{\mathcal{F}}(X^n)[X^{n+1} - X^n] + \mathcal{F}(X^n),$$

where $J_{\mathcal{F}}(X^n)$ is the jacobian of $\mathcal{F}(X)$ evaluated at X^n. Knowing the zero value of $\mathcal{F}$ in X^{n+1}, we then write:

$$J_{\mathcal{F}}(X^n)[X^{n+1} - X^n] = -\mathcal{F}(X^n).$$

Thus by putting

$$\mathbf{x} := X^{n+1} - X^n, \qquad \mathbf{A} := J_{\mathcal{F}}(X^n), \qquad \mathbf{b} := -\mathcal{F}(X^n).$$

we have a linear system to solve which is written:

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b} \tag{2}$$

Linear solvers

To solve linear systems, three approaches can be followed. The first one relies on using direct solve methods [5] in order to get an exact solution (at least in finite precision computations), the second consists in applying iterative methods which converge towards an approximate solution and the third is hybrid and is a mix of the two.
A direct method makes it possible to compute the solution of the system in a finite number of operations and in exact arithmetics. Some of the most known direct methods are:

- LU decomposition method [138]: consists in factorizing the system matrix $\mathbf{A}$ into a product of two triangular matrices (an upper one and a lower one). The solution of the linear system is then tantamount to solve two triangular systems successively by simple substitution.
- Cholesky method [35]: is a particular case of LU decomposition method when the matrix $\mathbf{A}$ is symmetric positive definite. The lower triangular matrix is the transpose of the upper triangular one.
- QR factorization method [60]: The idea is still to reach the solution of a triangular linear system. Yet, in this case the matrix $\mathbf{A}$ is factored as the product of a triangular matrix $\mathbf{Q}$ and of an orthogonal matrix $\mathbf{Q}$ ($\mathbf{Q}^{-1} = \mathbf{Q}^T$). The solution of the initial system $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ is equivalent to solving a reduced triangular system : $\mathbf{R} \cdot \mathbf{x} = \mathbf{Q}^T \cdot \mathbf{b}$.

Some of the advantages of such methods are robustness and precision. In fact, they don't depend on the sparsity or the conditioning of linear systems. Yet, most of those algorithms require in general $\mathcal{O}(n^3)$ operations to factor dense $n \times n$ matrices. In 1969, Strassen introduced the first algorithm [133] for matrix multiplication whose complexity is less than the conventional $\mathcal{O}(n^3)$ times, where the size of the matrices is $n \times n$. With this faster method, Strassen derived an efficient matrix inversion process of the same complexity: $\mathcal{O}(n^{log_2 7})$. However, subsequent studies showed that Strassen's algorithm has less favourable stability properties than conventional methods as it is only weakly stable [77]. In 1990, Coppersmith and Winograd improved the asymptotic complexity of matrix multiplication, down to $\mathcal{O}(n^{2.376})$ thus enabling a faster matrix inversion [40]. This barrier was broken in the last decade with the design of optimized CW-like algorithms running in time $\mathcal{O}(n^{2.3728639})$ [94]. On the other hand, this kind of methods requires a great amount of memory. Added to this is that computing an exact solution is time-consuming. For this reason, iterative methods are usually preferable. Furthermore, iterative linear solvers and preconditioners have been subject of research in the last decades and various iterative solution and preconditioning methods have been developed.

Methods of searching a solution in projection subspaces have been initiated by Krylov [90]. They aim at finding a solution to a linear system $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ of dimension n in a subspace of lower dimensions $m < n$ called search subspace. In a Krylov subspace method, an approximate solution $\mathbf{x}^{(m)}$ is sought from an affine subspace $\mathbf{x}^{(0)} + \mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)})$ by imposing that the residual $\mathbf{r}^{(m)}$ stays orthogonal to a subspace $\mathcal{L}_m$ of dimension m called subspace of constraints. The subspace $\mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)})$ is defined as the set containing of all linear combinations of vectors $\mathbf{r}^{(0)}, \mathbf{A} \cdot \mathbf{r}^{(0)}, \ldots, \mathbf{A}^{m-1} \cdot \mathbf{r}^{(0)}$, where $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(0)}$ is the residual for $\mathbf{x}^{(0)}$ the initial guess to the solution.

Among Krylov subspace solvers, we can cite the Conjugate Gradient (CG) method for linear systems with symmetric positive definite coefficient matrices [76], BiConjugate Gradient (BiCG) method [93], Conjugate Gradient Squared (CGS) method [130], BiConjugate Gradient Stabilized (BiCGStab) method [140], Orthomin method [142], Generalized Minimal RESidual (GMRES)

method [125] and Left Conjugate Direction (LCD) method [43] for non-symmetric systems.

CG method minimizes the $\mathbf{A}$-norm of the error, i.e. the $\mathbf{A}$-norm of the difference between the exact solution of the linear system and the approximate solution computed by the iterative solver is decreasing over iterations. For this method, the subspace of constraints is taken as $\mathcal{L}_m = \mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)})$. As a consequence, the residual vectors are orthogonal and this property yields short recurrence formulations (two-term or three-term) for CG method. The standard two-term formulation of CG is presented in Algorithm 1. Another feature of this method is its superlinear convergence [99] in $\left(\frac{\sqrt{\mathcal{K}(\mathbf{A})}-1}{\sqrt{\mathcal{K}(\mathbf{A})}+1}\right)^k$; k index designating the iterate, and $\mathcal{K}(\mathbf{A})$ the condition number of $\mathbf{A}$.

Algorithm 1 Conjugate Gradient method

Inputs: Coefficient matrix $\mathbf{A}$, right hand side $\mathbf{b}$, initial guess $\mathbf{x}_0$.
Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A} \cdot \mathbf{x}_0$ and set $\mathbf{p}_0 := \mathbf{r}_0$
$j := 0$
while (non-convergence)
  $\alpha_j := (\mathbf{r}_j^{\mathrm{T}} \mathbf{r}_j)/(\mathbf{p}_j^{\mathrm{T}} \mathbf{A} \cdot \mathbf{p}_j)$
  $\mathbf{x}_{j+1} := \mathbf{x}_j + \alpha_j \mathbf{p}_j$
  $\mathbf{r}_{j+1} := \mathbf{r}_j - \alpha_j \mathbf{A} \cdot \mathbf{p}_j$
  $\beta_j := (\mathbf{r}_{j+1}^{\mathrm{T}} \mathbf{r}_{j+1})/(\mathbf{r}_j^{\mathrm{T}} \mathbf{r}_j)$
  $\mathbf{p}_{j+1} := \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j$
end while

Similar to CG, the BiConjugate Gradient (BiCG) algorithm [93, 123] can be derived as projection process onto $\mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)})$ orthogonally to $\mathcal{L}_m = \mathcal{K}_m(\mathbf{A}, \check{\mathbf{r}}^{(0)})$ where $\check{\mathbf{r}}^{(0)}$ is an arbitrary vector satisfying $(\check{\mathbf{r}}^{(0)})^{\mathrm{T}} \mathbf{r}^{(0)} \neq 0$. Algorithm 2 presents the BiCG method.

Algorithm 2 BiConjugate Gradient method

Inputs: Coefficient matrix $\mathbf{A}$, right hand side $\mathbf{b}$, initial guess $\mathbf{x}_0$, initial vector $\check{\mathbf{r}}_0$ such that $(\check{\mathbf{r}}^{(0)})^{\mathrm{T}} \mathbf{r}^{(0)} \neq 0$.
Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A} \cdot \mathbf{x}_0$ and set $\mathbf{p}_0 := \mathbf{r}_0$, $\check{\mathbf{p}}_0 := \check{\mathbf{r}}_0$
$j := 0$
while (non-convergence)
  $\alpha_j := (\mathbf{r}_j^{\mathrm{T}} \check{\mathbf{r}}_j)/(\check{\mathbf{p}}_j^{\mathrm{T}} \mathbf{A} \cdot \mathbf{p}_j)$
  $\mathbf{x}_{j+1} := \mathbf{x}_j + \alpha_j \mathbf{p}_j$
  $\mathbf{r}_{j+1} := \mathbf{r}_j - \alpha_j \mathbf{A} \cdot \mathbf{p}_j$
  $\check{\mathbf{r}}_{j+1} := \check{\mathbf{r}}_j - \alpha_j \mathbf{A}^{\mathrm{T}} \cdot \check{\mathbf{p}}_j$
  $\beta_j := (\mathbf{r}_{j+1}^{\mathrm{T}} \check{\mathbf{r}}_{j+1})/(\mathbf{r}_j^{\mathrm{T}} \check{\mathbf{r}}_j)$
  $\mathbf{p}_{j+1} := \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j$
  $\check{\mathbf{p}}_{j+1} := \check{\mathbf{r}}_{j+1} + \beta_j \check{\mathbf{p}}_j$
end while

The Conjugate Gradient Squared (CGS) method [130, 123] was devised as a transpose-free variant of BiCG, i.e. it avoids computing matrix-vector product with $\mathbf{A}^{\mathrm{T}}$. For this purpose, it exploits the fact that the residual vectors $\mathbf{r}^{(m)}$ (resp. $\check{\mathbf{r}}^{(m)}$) can be expressed as matrix-vector products of polynomials of $\mathbf{A}$ (resp. $\mathbf{A}^{\mathrm{T}}$) by the initial residual vector $\mathbf{r}^{(0)}$ (resp. $\check{\mathbf{r}}^{(0)}$) and replaces the products of polynomials of $\mathbf{A}$ with polynomials of $\mathbf{A}^{\mathrm{T}}$ by squared polynomials of $\mathbf{A}$ in the inner product formulas of the step sizes $\alpha^{(j)}$ and $\beta^{(j)}$. Algorithm 3 presents the CGS method.

Algorithm 3 Conjugate Gradient Squared method

Inputs: Coefficient matrix $\mathbf{A}$, right hand side $\mathbf{b}$, initial guess $\mathbf{x}_0$, initial vector $\check{\mathbf{r}}_0$ such that $(\check{\mathbf{r}}^{(0)})^{\mathrm{T}}\mathbf{r}^{(0)} \neq 0$.

Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A} \cdot \mathbf{x}_0$ and set $\mathbf{p}_0 := \mathbf{r}_0, \mathbf{w}_0 := \mathbf{r}_0$
$j := 0$
while (non-convergence)
  $\alpha_j := (\mathbf{r}_j^{\mathrm{T}}\check{\mathbf{r}}_0)/(\check{\mathbf{r}}_0^{\mathrm{T}}\mathbf{A} \cdot \mathbf{p}_j)$
  $\mathbf{v}_j := \mathbf{w}_j - \alpha_j \mathbf{A} \cdot \mathbf{p}_j$
  $\mathbf{x}_{j+1} := \mathbf{x}_j + \alpha_j(\mathbf{v}_j + \mathbf{w}_j)$
  $\mathbf{r}_{j+1} := \mathbf{r}_j - \alpha_j \mathbf{A} \cdot (\mathbf{v}_j + \mathbf{w}_j)$
  $\beta_j := (\mathbf{r}_{j+1}^{\mathrm{T}}\check{\mathbf{r}}_0)/(\mathbf{r}_j^{\mathrm{T}}\check{\mathbf{r}}_0)$
  $\mathbf{w}_{j+1} := \mathbf{r}_{j+1} + \beta_j \mathbf{v}_j$
  $\mathbf{p}_{j+1} := \mathbf{w}_{j+1} + \beta_j(\mathbf{v}_j + \beta_j \mathbf{p}_j)$
end while

The BiConjugate Gradient STABilized (BiCGStab) method [140, 123] was developed to avoid the irregular convergence patterns of the CGS method due to the squaring of the residual polynomials. In fact, it replaces the squared residual polynomial in CGS by a product of this latter polynomial by a new one that is defined recursively to smooth the convergence of the algorithm. At each iteration k of BiCGStab, the degree of this new polynomial is equal to k. Algorithm 4 presents the BiCGStab method.

Algorithm 4 BiConjugate Gradient STABilized method

Inputs: Coefficient matrix $\mathbf{A}$, right hand side $\mathbf{b}$, initial guess $\mathbf{x}_0$, initial vector $\check{\mathbf{r}}_0$ such that $(\check{\mathbf{r}}^{(0)})^{\mathrm{T}}\mathbf{r}^{(0)} \neq 0$.

Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A} \cdot \mathbf{x}_0$ and set $\mathbf{p}_0 := \mathbf{r}_0$
$j := 0$
while (non-convergence)
  $\alpha_j := (\mathbf{r}_j^{\mathrm{T}}\check{\mathbf{r}}_0)/(\check{\mathbf{r}}_0^{\mathrm{T}}\mathbf{A} \cdot \mathbf{p}_j)$
  $\mathbf{v}_j := \mathbf{r}_j - \alpha_j \mathbf{A} \cdot \mathbf{p}_j$
  $\gamma_j := (\mathbf{v}_j^{\mathrm{T}}\mathbf{A} \cdot \mathbf{v}_j)/((\mathbf{A} \cdot \mathbf{v}_j)^{\mathrm{T}}(\mathbf{A} \cdot \mathbf{v}_j))$
  $\mathbf{x}_{j+1} := \mathbf{x}_j + \alpha_j \mathbf{p}_j + \gamma_j \mathbf{v}_j)$
  $\mathbf{r}_{j+1} := \mathbf{v}_j - \gamma_j \mathbf{A} \cdot \mathbf{v}_j$
  $\beta_j := \alpha_j(\mathbf{r}_{j+1}^{\mathrm{T}}\check{\mathbf{r}}_0)/(\gamma_j \mathbf{r}_j^{\mathrm{T}}\check{\mathbf{r}}_0)$
  $\mathbf{p}_{j+1} := \mathbf{r}_{j+1} + \beta_j(\mathbf{p}_j - \gamma_j \mathbf{A} \cdot \mathbf{p}_j)$
end while

GMRES method [125, 123] is a linear solver for general use which is largely used for non-symmetric linear systems. It is a Krylov subspace method where the subspace of constraints is taken as $\mathcal{L}_m = \mathbf{A}\mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)})$. Furthermore, there is a special feature of GMRES method: The kth iterate minimizes the euclidean norm of the residual in the Krylov space based on k vectors, and as every subspace is included in the subsequent one, the residual decreases monotonously. Algorithm 5 describes the generalized minimal residual method.

We can also cite multigrid methods [26] as another type of iterative solution methods. Multigrid solvers, comprising algebraic and geometric multigrid solvers, allow to obtain a solution of the linear system by combining a classic iterative method on the initial mesh with a correction method on coarse meshes. Algebraic multigrid methods (AMG) have proven to be the most efficient in the case of linear systems resulting from elliptic or parabolic problems [135].

Algorithm 5 GMRES method

Inputs: Coefficient matrix $\mathbf{A}$, right hand side $\mathbf{b}$, initial guess $\mathbf{x}_0$, Krylov subspace dimension m, $(m+1) \times m$ Hessenberg matrix $\overline{\mathbf{H}}_m$ of coefficients $(\mathbf{H}_{i,j})_{1 \leq i \leq m+1, 1 \leq j \leq m}$ initialized to $\mathbf{0}$.

1: Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A} \cdot \mathbf{x}_0$, $\beta := ||\mathbf{r}_0||_2$ and $\mathbf{v}_1 := \beta^{-1}\mathbf{r}_0$
2: $j := 1$
3: **while** $(j < m)$
4: Compute $\mathbf{w} := \mathbf{A} \cdot \mathbf{v}_j$
5: **for** $i = 1 \ldots j$ **Do**
6: $\mathbf{H}_{i,j} := \mathbf{w}^{\mathrm{T}}\mathbf{v}_i$
7: $\mathbf{w} := \mathbf{w} - \mathbf{H}_{i,j}\mathbf{v}_i$
8: **end for**
9: $\mathbf{H}_{j+1,j} := ||\mathbf{w}||_2$
10: **if** $(\mathbf{H}_{j+1,j} \neq 0)$
11: $\mathbf{v}_{j+1} := \mathbf{H}_{j+1,j}^{-1}\mathbf{w}$
12: **else**
13: Set $m = j$ and go to line 17
14: **end if**
15: $j := j + 1$
16: **end while**
17: Compute $\mathbf{y}_m$ that minimizes $||\beta\mathbf{e}_1 - \overline{\mathbf{H}}_m \cdot \mathbf{y}||_2$, with $\mathbf{e}_1 = (1, 0, \ldots, 0)^{\mathrm{T}}$
18: Update the initial guess $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{V}_m \cdot \mathbf{y}_m$ where $\mathbf{V}_m = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$

Domain decomposition [47] can generally be considered as a numerical technique applied to the solution of linear systems that combines both iterative and direct methods. In fact, the problem is partitioned between several subdomains on which the reduced systems are solved via direct solvers. The convergence of the global system is obtained by iterating and exchanging data between subdomains. On the whole, domain decomposition methods are an efficient means of solving various physical and mathematical problems. They have the advantage of providing a rigorous framework from a mathematical point of view to couple different types of equations, geometries and meshing.

Preconditioning the linear systems

Moreover, as iterative methods based on Krylov subspaces rely on the construction of a base from a sequence of successive matrix powers multiplied by the initial residual, it is evident that they will converge after N iterations, where N is the size of the system. However, in case of rounding errors, this characteristic is no longer always satisfied. Besides, in practice N may be very large and the iterative process can reach a sufficient precision much before in simple cases.

The convergence speed of iterative methods depends on certain characteristics of the system's matrix, especially on the matrix's conditioning and spectrum. The conditioning of the matrix $\mathbf{A}$ measures the linear system's sensitivity to the perturbation of the data $\mathbf{A}$ or $\mathbf{b}$. For a given vector norm $||\bullet||$, it is defined as follows:

$$\mathcal{K}(\mathbf{A}) = ||\mathbf{A}|| * ||\mathbf{A}^{-1}|| \quad ; \quad ||\mathbf{A}|| := \sup_{\mathbf{x} \neq \mathbf{0}} \frac{||\mathbf{A}\mathbf{x}||}{||\mathbf{x}||}. \tag{3}$$

The convergence can be very slow in case the system is ill-conditioned: the highest the conditioning is, the most digits of accuracy will be lost during the computation [34].

For this reason, the usage of a preconditioner should be considered in order to reduce the conditioning of the matrix and by consequence reduce the number of iterations so as to reach convergence. It is a technique that consists in transforming the initial problem $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ into an equivalent problem $\mathbf{M}^{-1}\mathbf{A} \cdot \mathbf{x} = \mathbf{M}^{-1} \cdot \mathbf{b}$ which can be more easily solved (because the conditioning of $\mathbf{M}^{-1}\mathbf{A}$ is less than that of $\mathbf{A}$) by multiplying by a matrix (called preconditioner).

That being said, preconditioning may not be sufficient to speed-up the convergence of a linear system, because by preconditioning we manage to reduce the conditioning, which is equivalent to a ratio between maximum and minimum singular values in the case of a Frobenius norm. Yet, the smallest singular values remain sometimes very low. As a result, the convergence is affected and numerically we do not get the same behaviour as in theory.

Various types of preconditioners have been suggested in the literature covering the broad field numerical linear algebra. These fall into different categories. The preconditioners can be classified according to their construction strategy and are broadly subdivided into three main groups: implicit, explicit and hybrid preconditioners. Implicit preconditioning methods target the construction of $\mathbf{M}$ while explicit methods aim at forming $\mathbf{M}^{-1}$. Preconditioners based on fixed-point iteration and incomplete factorization preconditioners [123] are two well known examples of implicit methods. The former are obtained by applying a decomposition strategy on the matrix. A fixed-point iteration preconditioner is obtained by applying a splitting strategy on the matrix. Of this type of preconditioners, we can cite the most commonly used ones: Jacobi, SOR and SSOR [123]. An incomplete factorization preconditioner is obtained by applying a factorization on the matrix (Gauss method) and respecting certain predefined fill-in rules. Thus, this avoids having factorization matrices that are denser than the initial matrix. We can mention two common examples of incomplete factorization methods: Incomplete LU decomposition (ILU) and Incomplete Cholesky decomposition (IC). The former is valid for general invertible matrices, whereas the latter necessitates symmetric positive definite matrices with some form of diagonal dominance but this difficulty can be circumvented by considering shifted matrices. We refer to [123] for more details on these decompositions and their different variants: ILU with minimal fill-in threshold (ILUT), ILU without fill-in (ILU(0) method is presented in Algorithm 6), ILU with level of fill p (ILU$(p)_{p=1,2,\ldots}$), IC without fill-in (IC(0)), etc... Still in the same class of preconditioners, we can mention preconditioners based on low rank approximation. This technique aims at approximating the singular value decomposition of a matrix (*cf.* [65] and references therein). When combined with a truncated factorization and applied on the unfactorized block which is a Schur complement, this technique yields LORASC preconditioners [67]. The main features of a LORASC preconditioner are that it allows to bound the condition number of the preconditioned matrix, and the fact that the preconditioner is fully algebraic.

Among explicit preconditioning methods, we can cite SPAI preconditioners in which we compute sparse approximate inverses [63, 70, 6, 21], polynomial preconditioners [98] and domain decomposition based preconditioners [27]. Polynomial preconditioning reduces the conditioning of the preconditioned system by introducing polynomials of the matrix as preconditioner. The principle is based on Cayley-Hamilton theorem according to which it holds for an invertible matrix $\mathbf{A}$ of size n that $\mathbf{A}^{-1}$ can then be written in the form of a polynomial of order $n-1$ evaluated at $\mathbf{A}$. Hence the appropriateness of seeking an approximation of $\mathbf{A}^{-1}$ by a polynomial in $\mathbf{A}$. Thus, this

Algorithm 6 Incomplete LU factorization without fill-in

Inputs: Coefficient matrix $\mathbf{A}$ of size $n \times n$, non zero pattern of the matrix $\mathrm{NZ}(\mathbf{A})$.

1: **for** $i = 2 \dots n$ **Do**
2: **for** $k = 1 \dots i-1$ such that $(i,k) \in \mathrm{NZ}(\mathbf{A})$ **Do**
3: Compute $a_{ik} := a_{ik} / a_{kk}$
4: **for** $j = k+1 \dots n$ such that $(i,j) \in \mathrm{NZ}(\mathbf{A})$ **Do**
5: Compute $a_{ij} := a_{ij} - a_{ik} a_{kj}$
6: **end for**
7: **end for**
8: **end for**

approximation of $\mathbf{A}^{-1}$ will form our preconditioner $\mathbf{M}^{-1}$ and consequently we will have $\mathbf{M}^{-1}\mathbf{A}$ which will be close to the identity matrix, and as a result its conditioning will be more reduced (ideally very close to 1): $\mathcal{K}(\mathbf{M}^{-1}\mathbf{A}) \ll \mathcal{K}(\mathbf{A})$. Domain decomposition methods have been the subject of advanced studies in recent years [116, 136, 47]. There are two types of applications of these methods: globally like a numerical technique applied to the solution of equations and locally as a method of preconditioning of linear systems. The success of this kind of methods lies in the fact that they allow for a high level of parallelism in addition to their easy implementation on most modern parallel computers [69]. Preconditioning strategies existing in the literature include Additive Schwarz (ASM), Restricted Additive Schwarz (RAS) and multilevel (DD) preconditioners. Many studies have been reported on the principle and properties of these preconditioners, for example in [136, 47, 27]. The approach relies on the partitioning of the initial domain into several subdomains. Restriction operators are defined to construct submatrices that are specific to each subdomain. Then, the reduced problems with these local submatrices are solved, we thereby obtain local preconditioners to which extension operators are applied. The global preconditioner is ultimately built by summing all these subdomain preconditioners. It should be noted that the subdomains are taken in such a way that there exists an overlap of variable size between them. One major difference between ASM and RAS is that for the first, the terms corresponding to the preconditioning on the overlapping zone are taken into account twice, while for the second, the extension operators are taken in such a way as to satisfy a partition of unity [47] on the overlapping zone. The use of this class of methods for problems stemming from porous media flow simulations has been covered in [74]. Another Domain Decomposition-based preconditioning strategy is the method of block preconditioning called Block-Jacobi *cf.* [123]. A Block-Jacobi preconditioner is defined by considering the diagonal blocks of the matrix. The Block-Jacobi method is the simplest form of domain decomposition. It can be considered as an Additive Schwarz without overlap between subdomains. A Block-Jacobi preconditioner is defined by considering the diagonal blocks $\mathbf{A}_{ii}$ of the matrix $\mathbf{A}$ where each block contains the coefficients specific to a subdomain i.

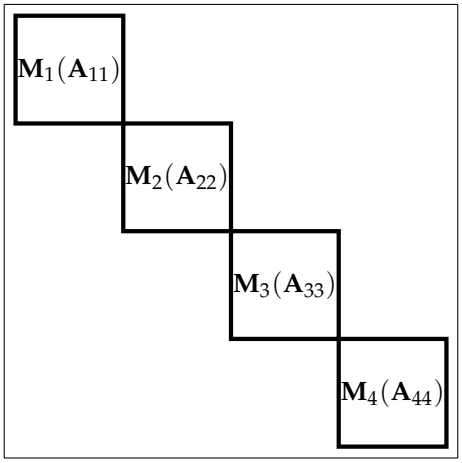


Figure 1 – *Example of Block-Jacobi preconditionner on* 4 *subdomains without overlap*

The application of a Block-Jacobi preconditioner is the simplest but also the most parallel. In fact, we can easily see that the Block-Jacobi preconditioner, as demonstrated in Figure 1 where $\mathbf{M}_i(\mathbf{A}_{ii})$ stands for the preconditioner chosen for the subdomain i, can be computed in parallel. Indeed, we only need to separately factor the diagonal blocks in parallel (see Figures 2 and 3).

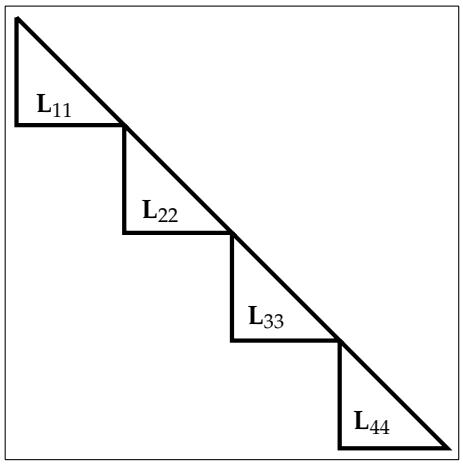


Figure 2 – *Lower triangular matrix* **L**

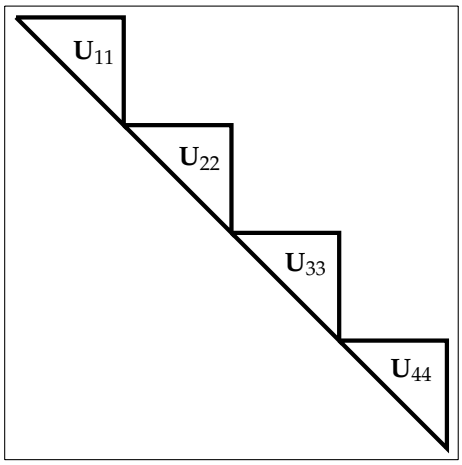


Figure 3 – *Upper triangular matrix* **U**

On the other hand, with such a disposition of blocks, the number of solver iterations tends to increase with the number of subdomains. To address this problem, some techniques can be used to allow the exchange of information between subdomains like overlapping or coarse grid correction for example, see [47] and references therein. It should also be added that for this strategy of block preconditioning, it is not really necessary to use the same preconditioner for all the blocks. It is possible to combine several preconditioners: For example, we can refer to [104], where a Block-Jacobi preconditioner is used on two blocks and various local preconditioning options (such as Jacobi, SSOR, FSAI* [146]) are chosen depending on the specificities of each block.

$*$. Factorized Sparse Approximate Inverse.

It should be noted that explicit preconditioners often necessitate a relatively more filled nonzero pattern (for the preconditioner to be constructed) to get a close approximation of the inverse, especially when $\mathbf{A}$ is ill-conditioned [12]. By contrast, implicit conditioners can provide a sufficiently reliable approximation of the inverse, if the matrix is sparse, even if the number of non-zeros in the preconditioner to be constructed is rather reduced.

There also exist combinations of these two categories of preconditioners that have been introduced *cf.* [139, 12, 146]: for example, a explicit preconditioning stage is performed first on matrix $\mathbf{A}$ yielding $\mathbf{M}_1$. This is followed by an implicit preconditioning stage on $\mathbf{I} - \mathbf{M}_1\mathbf{A}$, yielding $\mathbf{M}_2$. The final preconditioner obtained is implicit and is expressed: $\mathbf{M}_3 = \mathbf{M}_2^{-1}\mathbf{M}_1$. These hybrid preconditioners turn out to be more efficient than a simple implicit preconditioner, especially when matrix $\mathbf{I} - \mathbf{M}_1\mathbf{A}$ is more sparse than matrix $\mathbf{A}$. Nevertheless, for the hybrid preconditioner to be practically viable, preconditioner $\mathbf{M}_1$ should have a simple sparse structure (diagonal or tridiagonal for example). In the rest of this study, we are more interested in implicit preconditioners.

We have given so far an overview of general preconditioning methods. There also exist some application-specific preconditioners. In fact, to simulate a large-scale physical problem, preconditioners based on physical characteristics normally have a much better performance than ordinary preconditioners. Wallis and al. [145] have developed a family of Constraint Pressure Residual (CPR) preconditioners in order to accelerate the convergence of linear solvers for an oil model called "Black-Oil". This involves a two-stage preconditioning that combines a parallel AMG preconditioner for the pressure block of the linear system comprising elliptic equations, together with another parallel preconditioner based on an incomplete factorization (of ILU type) on the whole system. CPR-AMG [91, 127] is the state of the art of the preconditioners currently used in dynamic geoscience simulators [29, 135]. It is much more efficient than ILU preconditioners and it is very popular in the reservoir simulation field since it has a very good robustness with respect to data heterogeneity and anisotropy of media. A basic presentation of this two-stage preconditioner is given in Algorithm 7.

Algorithm 7 CPR-AMG Preconditioning

Inputs: pressure block coefficients $\mathbf{A}_{pp}$, non-pressure block coefficients $\mathbf{A}_{ss}$, coupling blocks $\mathbf{A}_{sp}$ and $\mathbf{A}_{ps}$, pressure unknowns $\mathbf{x}_p$, non-pressure unknowns $\mathbf{x}_s$, pressure right hand side $\mathbf{b}_p$, non-pressure right hand side $\mathbf{b}_s$,

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{pp} & \mathbf{A}_{ps} \\ \mathbf{A}_{sp} & \mathbf{A}_{ss} \end{pmatrix}, \quad \mathbf{x} = \begin{bmatrix} \mathbf{x}_p \\ \mathbf{x}_s \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_p \\ \mathbf{b}_s \end{bmatrix}.$$

1: Apply ILU(0) on the full system $\text{ILU0}(\mathbf{A}) \cdot \mathbf{x} = \mathbf{b}$
2: Compute the new residual $\mathbf{r}_p = \mathbf{b}_p - \mathbf{A}_{pp} \cdot \mathbf{x}_p^{(1)} - \mathbf{A}_{ps} \cdot \mathbf{x}_s^{(1)}$
3: Apply AMG-Vcycle on the residual pressure equation $\text{AMG}(\mathbf{A}_{pp}) \cdot \mathbf{x}_p^{(2)} = \mathbf{r}_p$
4: Correct the pressure unknowns $\mathbf{x}_p = \mathbf{x}_p^{(1)} + \mathbf{x}_p^{(2)}$

In brief, the preconditioning strategies are many and varied. On the other hand, nothing guarantees that there is an optimal solution for all cases of application, but what is evident is that each solution is adapted to a certain type of matrices for which it offers more advantages than drawbacks. Let us recall some criteria for judging the efficiency of a preconditioner:

- the spectrum of $\mathbf{M}^{-1}\mathbf{A}$;
- the accuracy, that we can measure by the norm $\|\mathbf{M}^{-1} \cdot \mathbf{A} - \mathbf{I}\|$ (Indeed, a preconditioner is expected to approximate the inverse of the system's matrix);

- the complexity, especially with the fill-in levels of preconditioners for sparse matrices: SPAI [63, 70] and $ILU(p)_{p=0,1,2,...}$ [123]. The less regular the sparsity pattern that we have preset for the preconditioner is, the more complex its construction will be;
- the computation time necessary to build the preconditioner;
- the scalability, i.e. how this preconditioner performs at large scale (on very large matrices), and if it is highly parallelizable.

Approximate inversion

Approximate inversion is a field in linear algebra that has been actively explored over the last decades mainly for deriving preconditioning techniques. There is an abundant literature on this subject [123, 63, 70, 6, 21]. Approximate inversion methods fall into two broad categories: factorized sparse approximate inverses and approximate inverse methods based on Frobenius norm minimization.

For the former category, we can mention methods such as incomplete LU factorization with its different variants (zero fill-in ILU, ILU with level of fill, modified ILU, ILU with threshold) [123]. This factorization computes sparse triangular matrices $\mathbf{L}$ and $\mathbf{U}$, respectively lower and upper, so that the residual matrix $\mathbf{R} := \mathbf{LU} - \mathbf{A}$ satisfies a particular sparsity pattern. This is achieved by dropping some elements outside of the main diagonal during the Gaussian Elimination process. Dropping an element means that this non-zero entry was transformed into a zero. The zero fill-in variant, denoted by ILU(0) is obtained when the chosen zero pattern is exactly the zero pattern of the matrix $\mathbf{A}$. The broader ILU(p) variant assigns a level of fill p to each element that is processed by Gaussian elimination, and computes the LU factorization for elements with a level of fill less than or equal to p. This allows to improve the accuracy of the factorization. The exact factorization of the matrix $\mathbf{A}$ is obtained if $p = n - 1$, where n is the size of $\mathbf{A}$. Then, the smaller p is taken, the less accurate and the quicker the factorization becomes. The modified ILU (denoted by MILU) variant tries to compensate for the entries that were dropped out in approximate factorizations by substracting them from the diagonal entry in $\mathbf{U}$. The ILU with threshold (denoted by ILUT) is based on discarding small elements during the Gaussian elimination. For example, dropping criteria could be set such that the non-zero entries whose magnitude is less than a fixed threshold are ignored.

Several other incomplete factorizations have been developed as well in the literature, e.g. incomplete Cholesky, incomplete $\mathbf{LDL}^{\mathrm{T}}$ factorization [123]. Those factorizations are computed by a modified Gaussian elimination where fill-ins are limited to certain matrix locations that are defined by the sparsity pattern. A variety of patterns have been explored, see [128] and references therein. Furthermore, much work has been done or initiated to address some efficiency issues in order to get a better parallelism [48, 59, 36].

On the other hand, for the second category of approximate inversion methods, we can cite sparse approximate inverses first proposed in the 1970s [57, 20]. These methods are based on approximating the matrix inverse directly, $\mathbf{M} \approx \mathbf{A}^{-1}$. This matrix is then explicitly computed and stored. It is applied by a matrix-vector product unlike the forementioned factorized approximate methods which require forward and backward substitution operations. The matrix $\mathbf{M}$ is basically computed as the solution of a constrained minimization problem: $\min_{\mathbf{M} \in \mathcal{S}} \|\mathbf{I} - \mathbf{AM}\|_F$ for a right-

approximate inverse; or as the solution of $\min_{\mathbf{M}\in\mathcal{S}} \|\mathbf{I}-\mathbf{MA}\|_F$ for a left-approximate inverse where $\mathcal{S}$ is a set of sparse matrices and $\|\ \|_F$ is the Frobenius norm. Various strategies have been derived for the choice of $\mathcal{S}$ [41, 63, 78]. The approach introduced in [70] resulted in the SPAI preconditioner. In an attempt to lower the computational cost and storage requirements of building such an inverse, the Minimal Residual algorithm was designed [37, 123].

When employed as preconditioners, many incomplete factorizations like ILU have scalability problems often due to low eigenvalues that hamper the convergence of the iterative solver. To remedy that, particular filtering preconditioners M have been proposed, see [2] and references therein. They are chosen such that $\mathbf{MV} = \mathbf{KV}$, where $\mathbf{V}$ is a set of vectors of the directions to be filtered, that is, $\mathbf{V}$ can be filled with eigenvectors of the targeted lowest eigenvalues. In that case, the corresponding eigenvalues will be shifted to 1 in the spectrum of $\mathbf{M}^{-1}\mathbf{A}$. This kind of spectral property is quite convenient, especially when it comes to very small eigenvalues, and is often desired. In this respect, we can cite the article [88] where similar low-rank transformations constructed at several cycles of the Arnoldi procedure were used to translate spread eigenvalues into the vicinity of 1. Still on the subject of low rank approximation, [51] discusses rank-one updates, which are suited for stationary methods for solving linear systems $\mathbf{A}\cdot\mathbf{x} = \mathbf{b}$, to improve an approximation $\mathbf{H}$ of the inverse of the main matrix $\mathbf{A}$ during iterations. The method proposed reduces singular values of $\mathbf{I}-\mathbf{AH}$ and accelerates the convergence. In [67], an algebraic preconditioner based on a low rank approximation technique was introduced. It is robust as it bounds the condition number of the preconditioned matrix $\mathbf{M}^{-1}\mathbf{A}$ by a user defined threshold.

Error estimators

When a mathematical problem is solved using a numerical approximation method, the computations are subject to error. This error is generally unknown because we simply do not have the exact solution. However, the question that arises is to know, at the end of a calculation, an upper bound of the norm of this error. The existence of such an upper bound, called estimator, would in fact allow to quantify the pertinence of the approximate solution having just been computed. In this case, the estimator is said to be a posteriori when the estimation phase can be achieved only during or after the solution phase. On the other hand, the estimator is said to be a priori if it can be determined before any calculation. This latter estimator is a theoretical upper bound which gives, for example, an idea about the order of convergence. Nonetheless, the a priori estimator generally involves undetermined constants, and cannot always be numerically computed, unlike the a posteriori estimator.

There are many categories of a posteriori error estimators: Explicit residual estimators, *cf.* [14, 141, 30], the equilibrated residual method, *cf.* [92, 4], estimators based on smoothed stresses *cf.* [148], functional a posteriori estimators *cf.* [108, 117], hierarchical estimators *cf.* [18], a posteriori geometrical error estimators *cf.* [31, 58], and equilibrated fluxes a posteriori error estimates *cf.* [115, 44, 101, 3]. It is this last class of a posteriori error estimators that is of concern to us in this thesis. The main advantage of equilibrated fluxes estimates is that they provide a fully computable upper bound of the global error that can be decomposed into several components that identify the different sources of error in the numerical solution *cf.* [46, 55]. The distinction between the different components of the error (in space, in time, of linearization, and algebraic) helps to derive

adaptive algorithms which ensure a reduction of calculations by imposing a stopping criterion for iterations (of linearization and of the algebraic solver), by applying a time step adjustment and a balancing of spatial and temporal error components [144].

Pioneering works were carried out with the primary objective of identifying the *algebraic error* during the iterative solution of a linear system, raised from a numerical approximation of partial differential equations [25, 17, 15, 18, 118, 119, 120, 111]. However, subsequent works using the theory of the a posteriori error estimates in order to derive rigorous upper bounds of the global error including algebraic error have later followed [11, 9, 105, 79, 55, 114]. More recent works were carried on deriving appropriate guaranteed upper bound directly on the algebraic error using equilibrated flux reconstructions [114, 106, 113].

To the best of our knowledge, almost no work has been done to this day on the use of a posteriori error estimates of the algebraic error in order to provide an adaptive choice of preconditioners used for the solution of sparse linear systems stemming from the numerical approximation of partial differential equations. The goal of this thesis is to fill the gap. We propose different approaches, one based on the fact that the error estimates can provide a reliable information about the global error norms and thus gives an idea of each system's complexity during the simulation. Then, we derive different variants of an a posteriori-based adaptive preconditioner that is built following the local distribution of the algebraic error estimate.

Background notions

Some properties of Block-Jacobi preconditioning

In this section, we recall some properties of Block-Jacobi preconditioners. We follow the theory developed in [12] and generalize the results to the case with m blocks.

Let the symmetric positive definite matrix $\mathbf{A}$ be partitioned into m blocks as follows: $\mathbf{A} = [\mathbf{A}_{ij}]$ for $i, j = 1, \ldots, m$. We denote by n_i the size of the diagonal block $\mathbf{A}_{ii}$, and by $(V_i)_{1 \leq i \leq n}$ the finite dimensional spaces consistent with the above block partition of $\mathbf{A}$. Let $\mathbf{M}$ be the Block-Jacobi preconditioner of $\mathbf{A}$: $\mathbf{M} = [\mathbf{M}_{ij}]$ for $i, j = 1, \ldots, m$ where:

$$\mathbf{M}_{ij} = \begin{cases} \mathbf{A}_{ii} & \text{if } i = j \\ \mathbf{0}_{ij} & \text{otherwise} \end{cases} \tag{4}$$

Let γ_{ij} be the Cauchy-Schwarz-Bunyakowski (C.B.S.) constant [12] defined for the 2×2 block matrix composed by $\mathbf{A}_{ii}$ and $\mathbf{A}_{jj}$ as diagonal blocks, $\mathbf{A}_{ij}$ and $\mathbf{A}_{ji}$ as off-diagonal blocks:

$$\gamma_{ij} := \sup_{\mathbf{v}_i \in V_i, \mathbf{v}_j \in V_j} \frac{\mathbf{v}_i^{\mathrm{T}} \mathbf{A}_{ij} \mathbf{v}_j}{\left(\mathbf{v}_i^{\mathrm{T}} \mathbf{A}_{ii} \mathbf{v}_i \mathbf{v}_j^{\mathrm{T}} \mathbf{A}_{jj} \mathbf{v}_j \right)^{\frac{1}{2}}} \tag{5}$$

Lemma 0.1. *Let λ be an eigenvalue of $\mathbf{M}^{-1}\mathbf{A}$. We have*

$$1 - \max_{1\leq i\leq m} \sum_{\substack{j=1\\ j\neq i}}^{m} \gamma_{ij} \leq \lambda \leq 1 + \max_{1\leq i\leq m} \sum_{\substack{j=1\\ j\neq i}}^{m} \gamma_{ij} \tag{6}$$

Proof. The extreme eigenvalues of $\mathbf{A}\cdot\mathbf{x} = \lambda\mathbf{M}\cdot\mathbf{x}$ are the extreme values of

$$\frac{\mathbf{x}^{\mathrm{T}}\cdot\mathbf{A}\cdot\mathbf{x}}{\mathbf{x}^{\mathrm{T}}\cdot\mathbf{M}\cdot\mathbf{x}} = \frac{\sum_{j=1}^{m}\mathbf{x}_j^{\mathrm{T}}\cdot\mathbf{A}_{jj}\cdot\mathbf{x}_j + \sum_{j=1}^{m}\sum_{\substack{i=1\\ i\neq j}}^{m}\mathbf{x}_j^{\mathrm{T}}\cdot\mathbf{A}_{ji}\cdot\mathbf{x}_i}{\sum_{j=1}^{m}\mathbf{x}_j^{\mathrm{T}}\cdot\mathbf{A}_{jj}\cdot\mathbf{x}_j}; \qquad \mathbf{x} = \begin{bmatrix}\mathbf{x}_1\\ \vdots\\ \mathbf{x}_m\end{bmatrix}$$

For each pair of distinct indices (i,j), we have:

$$|\mathbf{x}_j^{\mathrm{T}}\cdot\mathbf{A}_{ji}\cdot\mathbf{x}_i| \leq \gamma_{ij}\left(\mathbf{x}_i^{\mathrm{T}}\mathbf{A}_{ii}\mathbf{x}_i\mathbf{x}_j^{\mathrm{T}}\mathbf{A}_{jj}\mathbf{x}_j\right)^{\frac{1}{2}} \leq \frac{\gamma_{ij}}{2}(\mathbf{x}_j^{\mathrm{T}}\cdot\mathbf{A}_{jj}\cdot\mathbf{x}_j + \mathbf{x}_i^{\mathrm{T}}\cdot\mathbf{A}_{ii}\cdot\mathbf{x}_i)$$

Therefore

$$-\sum_{j=1}^{m}\sum_{\substack{i=1\\ i\neq j}}^{m}\frac{\gamma_{ji}}{2}(\mathbf{x}_j^{\mathrm{T}}\cdot\mathbf{A}_{jj}\cdot\mathbf{x}_j + \mathbf{x}_i^{\mathrm{T}}\cdot\mathbf{A}_{ii}\cdot\mathbf{x}_i) \leq \mathbf{x}^{\mathrm{T}}\cdot\mathbf{A}\cdot\mathbf{x} - \sum_{j=1}^{m}\mathbf{x}_j^{\mathrm{T}}\cdot\mathbf{A}_{jj}\cdot\mathbf{x}_j \leq \sum_{j=1}^{m}\sum_{\substack{i=1\\ i\neq j}}^{m}\frac{\gamma_{ji}}{2}(\mathbf{x}_j^{\mathrm{T}}\cdot\mathbf{A}_{jj}\cdot\mathbf{x}_j + \mathbf{x}_i^{\mathrm{T}}\cdot\mathbf{A}_{ii}\cdot\mathbf{x}_i)$$

However, due to the symmetry we notice that:

$$\sum_{j=1}^{m}\sum_{\substack{i=1\\ i\neq j}}^{m}\frac{\gamma_{ji}}{2}(\mathbf{x}_j^{\mathrm{T}}\cdot\mathbf{A}_{jj}\cdot\mathbf{x}_j + \mathbf{x}_i^{\mathrm{T}}\cdot\mathbf{A}_{ii}\cdot\mathbf{x}_i) = \sum_{j=1}^{m}\mathbf{x}_j^{\mathrm{T}}\cdot\mathbf{A}_{jj}\cdot\mathbf{x}_j\sum_{\substack{i=1\\ i\neq j}}^{m}\gamma_{ji}$$

Consequently

$$\sum_{j=1}^{m}\mathbf{x}_j^{\mathrm{T}}\cdot\mathbf{A}_{jj}\cdot\mathbf{x}_j(1 - \sum_{\substack{i=1\\ i\neq j}}^{m}\gamma_{ji}) \leq \mathbf{x}^{\mathrm{T}}\cdot\mathbf{A}\cdot\mathbf{x} \leq \sum_{j=1}^{m}\mathbf{x}_j^{\mathrm{T}}\cdot\mathbf{A}_{jj}\cdot\mathbf{x}_j(1 + \sum_{\substack{i=1\\ i\neq j}}^{m}\gamma_{ji})$$

$$(1 - \max_{1\leq i\leq m}\sum_{\substack{j=1\\ j\neq i}}^{m}\gamma_{ij})\sum_{j=1}^{m}\mathbf{x}_j^{\mathrm{T}}\cdot\mathbf{A}_{jj}\cdot\mathbf{x}_j \leq \mathbf{x}^{\mathrm{T}}\cdot\mathbf{A}\cdot\mathbf{x} \leq (1 + \max_{1\leq i\leq m}\sum_{\substack{j=1\\ j\neq i}}^{m}\gamma_{ij})\sum_{j=1}^{m}\mathbf{x}_j^{\mathrm{T}}\cdot\mathbf{A}_{jj}\cdot\mathbf{x}_j$$

This latter inequality completes the proof. □

Here, we retrieve a known feature of Block-Jacobi preconditioners: they bound the maximum eigenvalue of the preconditioned matrix (see for example [89, 12, 103] and references therein). Indeed, according to Lemma 0.1, a Block-Jacobi preconditioner allows to keep the maximum eigenvalue of the preconditioned operator $\mathbf{M}^{-1}\mathbf{A}$ bounded by a constant that depends on the blocks of matrix. In fact, as the C.B.S. constants are less than or equal to 1 (because $\mathbf{A}$ is SPD [12]), we can deduce that the maximum eigenvalue is bounded by m the number of blocks.

Under the assumption that the value of $\max_{1\leq i\leq n}\sum_{\substack{j=1\\ j\neq i}}^{m}\gamma_{ij}$ is below 1 (which can somehow translate the block-dominance of the main block diagonal over the off-diagonal blocks), we can derive a bound for the condition number:

$$\mathcal{K}(\mathbf{M}^{-1}\mathbf{A}) \leq \left(1+\max_{1\leq i\leq n}\sum_{\substack{j=1\\ j\neq i}}^{m}\gamma_{ij}\right)\left(1-\max_{1\leq i\leq n}\sum_{\substack{j=1\\ j\neq i}}^{m}\gamma_{ij}\right)^{-1}.$$

In the case when approximate incomplete factorizations are used instead of exact ones for the diagonal blocks, we can obtain the following property.

Lemma 0.2. *Let the preconditioner $\widetilde{\mathbf{M}} = [\widetilde{\mathbf{M}}_{ij}]_{i,j=1,\ldots,m}$ be defined by:*

$$\widetilde{\mathbf{M}}_{ij} = \begin{cases} \widetilde{\mathbf{A}}_{ii} & \text{if } i=j \\ \mathbf{0}_{ij} & \text{otherwise} \end{cases} \qquad \text{with:} \quad \alpha_i^{(1)}\mathbf{A}_{ii} \leq \widetilde{\mathbf{A}}_{ii} \leq \alpha_i^{(0)}\mathbf{A}_{ii},\ \forall i \tag{7}$$

where the coefficients are such that $0 < \alpha_i^{(1)} \leq 1 \leq \alpha_i^{(0)}$, for all i and the inequalities are in a positive semidefinite sense, i.e.

$$\alpha_i^{(1)}\mathbf{x}_i^{\mathrm{T}}\cdot\mathbf{A}_{ii}\cdot\mathbf{x}_i \leq \mathbf{x}_i^{\mathrm{T}}\cdot\widetilde{\mathbf{A}}_{ii}\cdot\mathbf{x}_i \leq \alpha_i^{(0)}\mathbf{x}_i^{\mathrm{T}}\cdot\mathbf{A}_{ii}\cdot\mathbf{x}_i\ ,\ \forall \mathbf{x}_i \in V_i \tag{8}$$

Then we have:

$$\mathcal{K}(\widetilde{\mathbf{M}}^{-1}\mathbf{A}) \leq \mathcal{K}(\mathbf{M}^{-1}\mathbf{A}) \times \frac{\max\limits_{1\leq i\leq n}\alpha_i^{(0)}}{\min\limits_{1\leq j\leq n}\alpha_j^{(1)}} \tag{9}$$

Proof. In a similar way to the proof of Lemma 0.1, the extreme eigenvalues of $\widetilde{\mathbf{M}}^{-1}\mathbf{A}$ are the extreme values of

$$\frac{\mathbf{x}^{\mathrm{T}}\cdot\mathbf{A}\cdot\mathbf{x}}{\mathbf{x}^{\mathrm{T}}\cdot\widetilde{\mathbf{M}}\cdot\mathbf{x}} = \frac{\sum\limits_{j=1}^{m}\mathbf{x}_j^{\mathrm{T}}\cdot\mathbf{A}_{jj}\cdot\mathbf{x}_j + \sum\limits_{j=1}^{m}\sum\limits_{\substack{i=1\\ i\neq j}}^{m}\mathbf{x}_j^{\mathrm{T}}\cdot\mathbf{A}_{ji}\cdot\mathbf{x}_i}{\sum\limits_{j=1}^{m}\mathbf{x}_j^{\mathrm{T}}\cdot\widetilde{\mathbf{A}}_{jj}\cdot\mathbf{x}_j}$$

From (8), we have:

$$\min_{1\leq i\leq m}\alpha_i^{(1)}\sum_{j=1}^{m}\mathbf{x}_j^{\mathrm{T}}\cdot\mathbf{A}_{jj}\cdot\mathbf{x}_j \leq \sum_{j=1}^{m}\mathbf{x}_j^{\mathrm{T}}\cdot\widetilde{\mathbf{A}}_{jj}\cdot\mathbf{x}_j \leq \max_{1\leq i\leq m}\alpha_i^{(0)}\sum_{j=1}^{m}\mathbf{x}_j^{\mathrm{T}}\cdot\mathbf{A}_{jj}\cdot\mathbf{x}_j$$

It follows from this latter inequality that:

$$\frac{1}{\max\limits_{1\leq i\leq m}\alpha_i^{(0)}} \times \frac{\mathbf{x}^{\mathrm{T}}\cdot\mathbf{A}\cdot\mathbf{x}}{\mathbf{x}^{\mathrm{T}}\cdot\mathbf{M}\cdot\mathbf{x}} \leq \frac{\mathbf{x}^{\mathrm{T}}\cdot\mathbf{A}\cdot\mathbf{x}}{\mathbf{x}^{\mathrm{T}}\cdot\widetilde{\mathbf{M}}\cdot\mathbf{x}} \leq \frac{1}{\min\limits_{1\leq i\leq m}\alpha_i^{(1)}} \times \frac{\mathbf{x}^{\mathrm{T}}\cdot\mathbf{A}\cdot\mathbf{x}}{\mathbf{x}^{\mathrm{T}}\cdot\mathbf{M}\cdot\mathbf{x}}$$

Thus, we can deduce inequalities on the respective extreme eigenvalues of $\widetilde{\mathbf{M}}^{-1}\mathbf{A}$ and $\mathbf{M}^{-1}\mathbf{A}$ which complete the proof:

$$\lambda_{\min}(\widetilde{\mathbf{M}}^{-1}\mathbf{A}) \geq \frac{\lambda_{\min}(\mathbf{M}^{-1}\mathbf{A})}{\max\limits_{1\leq i\leq m} \alpha_i^{(0)}} \qquad \text{and} \qquad \lambda_{\max}(\widetilde{\mathbf{M}}^{-1}\mathbf{A}) \leq \frac{\lambda_{\max}(\mathbf{M}^{-1}\mathbf{A})}{\min\limits_{1\leq i\leq m} \alpha_i^{(1)}}$$

□

For more explanation on the inequalities in (7) and the existence of the coefficients $\alpha_i^{(0)}$ and $\alpha_i^{(1)}$ in the assumption of Lemma 0.2, see Remark 3.2 in Chapter 3.

1

Global and local adaptive preconditioning based on a posteriori error estimates for dynamic geoscience simulations

Contents

Abstract

This chapter is divided in two parts. In the first part, we propose a generic approach to choose or switch the preconditioner used to solve the linear systems in a simulation from a given range of options. The algorithm is adaptive in the sense that the choices are driven by the use of the global a posteriori estimation of the algebraic error, and made for the preconditioner to adjust to the matrix handled at each time step of a simulation. The approach is generic in the sense that it is not dependent on a specific class of error estimates. Numerical tests with real data sets taken from reservoir simulations show that with this approach, we can get some gain in time and efficiency as the robust (and costly to compute) preconditioners are employed only when needed. In the second part, we introduce an adaptive preconditioning approach based on the local a posteriori estimation of the algebraic error. In the suggested method, the matrix is block-wise treated: we propose a Block-Jacobi preconditioner assembled from local preconditioners computed distinctly for the diagonal blocks according to the error estimates values on each corresponding subdomain. In fact, we distinguish the subdomains with large error estimates values (HE subdomains), and the subdomains where the error estimates are low or negligible (LE subomdains). This way, it is possible to employ more robust preconditioners, even if they are more computationally costly, for matrix blocks that correspond to the HE subdomains and less robust, less costly preconditioners for the matrix blocks corresponding to the LE subdomains. Numerical tests with real data sets taken from reservoir simulations confirm that computational gains can be achieved thanks to this approach.
Keywords— Choice of preconditioners, error estimates, adaptivity, iterative solve, domain decomposition, matrix partitioning, block diagonal preconditioning

1.1 Global adaptive preconditioning based on a posteriori error estimates

A posteriori error estimators are considered to be an efficient means to control non linear algebraic systems arising from the discretization of PDEs [55]. As it has been mentioned in the previous chapter, the a posteriori flux balancing error estimation that we are concerned with can consist of several components of various kinds depending on their sources [55]. In this respect, we could mention the algebraic error estimate, the linearization error estimate, the discretization error estimate (or spatial estimate), etc ... All these error estimates are computed throughout the simulation, either globally on the whole mesh, or locally for each mesh element, at the rate of one spatial estimate and one algebraic estimate per time step.

This type of estimates has already been used to establish stopping criteria for the solution of non-linear PDEs stemming from dynamic simulations in Geosciences [45]. In porous media flow simuations, the nature of the model * and the way the equations are solved determine the size of the linearized system (2), namely the size of the Jacobian matrix $\mathbf{A}$, and the length of the vector of unknowns $\mathbf{x}$. In this chapter, also in Chapter 5 we use the linear systems stemming from single phase, two-phase, and three-phase flow problems. In the BlackOil model that we consider in these three chapters, we have three unknowns for the triphasic case, two unknowns for the two-phase and one unknown for the single-phase case.

The distinction made between error components leads to stopping criteria that show that there is

*. flow models from single-phase to tri-phasic; BlackOil/compositional; in simple or double media (fractured reservoir)

no need to pursue iterations of the linear solver once the other error components (of discretization and linearization) start to become dominant. Similarly, it is needless to continue iterations on the nonlinear solver when the discretization error component is dominant *cf.* [55, 46]. The idea is to introduce thresholds for ratios between error components. For example, we are going to use scalars that we will note γ_{alg} and γ_{lin} to compare the algebraic error estimate and the linearization error estimate respectively with the global error estimate and therefore determine when exactly one of them does not affect the global error.

Among the error estimators cited above, we are going to focus on spatial and algebraic error estimators. The obtained approximate solution is considered satisfactory if the algebraic error η_{alg} represents a small percentage of the partial global error η_{sp}:

$$\eta_{\text{alg}} < \gamma_{\text{alg}} * \eta_{\text{sp}} \tag{1.1}$$

The definition of γ_{alg} depends on the desired precision, its value is typically of order 0.1 [55]. Here, we would like to point out that this work builds on error estimates, such as the a posteriori flux balancing estimates cited earlier and that we used in practice in the numerical section of this chapter, but is not restricted to this particular family of estimates. The adaptive approach suggested in this thesis is compatible with a broad range of estimates. For more details on the derivation of some error estimates, we refer to Chapter 2 (*cf.* Section 2.3) where we give a brief explanation of how to compute a posteriori flux balancing error estimates in a finite element framework.

This chapter is organized as follows. In Section 1.1.1, we introduce the concept of adaptive choice of preconditioning in a simulation by using error estimators a posteriori. Then, the adaptive preconditioning algorithm is presented in Section 1.1.2. Subsequently, the execution plateform, the computing framework and the study cases are covered in Sections 1.1.3, 1.1.4 and 1.1.5 respectively. We use a preconditioned BiCGStab and a preconditioned GMRES to evaluate the efficiency of the choice of a preconditioner influenced by a posteriori error estimators. We will analyze the results using BiCGStab in Section 1.1.6 and GMRES in Section 1.1.7. Then, the remainder of the chapter deals with a local adaptive preconditioning strategy. In Section 1.2.1, we investigate a link between a posteriori error estimates and the conditioning of the matrices. Section 1.2.2 focuses on the partitioning of the matrix based on a posteriori error estimates. Section 1.2.3 defines the concept of variable Block-Jacobi preconditioning according to error estimates. Finally, the efficiency of the method is assessed with the numerical tests of Section 1.2.4.

1.1.1 Adaptative choice of preconditioners from a posteriori error estimates in a simulation

In this part, we intend to introduce preconditioners' choice criteria with a posteriori spatial and algebraic error estimates. Why did we choose these two estimates in particular? Because the first includes the sum of error components and the second represents the error component relating to the iterations of the algebraic solver, which can have a possible link with the preconditioning quality of the latter. At this stage of study, we are not yet able to prove either theoretically or experimentally the nature of the link between the computed error estimators and the conditioning of the matrix. In the rest of this chapter, we will suggest a heuristic of preconditioners choice whose results will be verified experimentally. In fact, in the same way as for stopping criteria,

we would like to use a relation similar to (1.1) to choose a preconditioner for a given system. For this, as it has been mentioned earlier, a posteriori error estimators are computed throughout the simulation, either globally or locally on each mesh point or node. The idea behind this approach is to examine the feasibility of global a posteriori error estimators' usage to choose a preconditioner for the different systems obtained in the course of the simulation. The aim is to adapt the preconditioner on the basis of a posteriori error estimators that have been computed instead of having one fixed preconditioner during the whole simulation for all the systems.

To be more precise, we would like to choose between two classes of preconditioners, on the basis of the values of error estimators. Preconditioners $\mathbf{M}_S$ of the first class are more robust than those of the second class $\mathbf{M}_W$, but they can be costlier in terms of memory. By robustness, we mean the ability to solve a wide range of problems at least reasonably well. In case the algebraic error estimator η_{alg} represents a small percentage of the spatial error η_{sp}, i.e. below a certain threshold γ_{prec},

$$\eta_{alg} \leq \gamma_{prec} * \eta_{sp}, \tag{1.2}$$

a preconditioner $\mathbf{M}_W$ is chosen. In the opposite case, it is a preconditioner $\mathbf{M}_S$ which is selected.

It is worth noting that the efficiency of such selection procedure depends on the preconditioners $\mathbf{M}_W$ and $\mathbf{M}_S$ that the user provides. Equation (1.2) simply gives an estimation of the ratio of the algebraic error in comparison with the global error obtained in the simulation, but it doesn't give information about the conditioning of the systems. Thus, the user has to provide two preconditioners where at least the systems preconditioned by $\mathbf{M}_S$ would converge. This being said, the algorithm generalizes easily to more than two preconditioners. Only an order of use for all possible preconditioners needs to be specified.

In the sequel, we introduce the different possible options for the preconditioner's choice (Section 1.1.2), and check their effect on the convergence in Section 1.1.6. In Section 1.1.7, we examine the performance of our adaptive algorithms of preconditioner choice with a linear solver that integrates well with restarting, GMRES.

1.1.2 General parameters

Apart from the computation of the algebraic error estimator (η_{alg}), there is a number of parameters to be taken into consideration before selecting the preconditioner to be used. Firstly, do we have to choose a preconditioner for every system of the simulation, or is this choice made only for certain systems during the simulation and maintained for subsequent systems? Secondly, at what iteration is the algebraic error estimator η_{alg} to be evaluated? Thirdly, are we supposed to take into account the solution x_f obtained in the computing process of η_{alg} and reuse it as a starting point x_0 for the preconditioned solver even if some of these solvers, such as BiCGStab are not inherently restartable? Or should we consider the computing of η_{alg} as a pre-processing stage and initialize x_0 to 0 for the preconditioned solver?

At every time step, we need to solve at least one linear system obtained in one Newton iteration. In practice, we assume that the linear systems obtained during the same time step have similar condition numbers. This is, somehow, in the same line with the decision to assess the algebraic estimator once per time step for reasons of computational savings. Thus, during every time step of the simulation, it will be possible to choose a preconditioner for the first linear system obtained

from the first Newton step, and to use this same preconditioner for the rest of the systems obtained during this time step.

Remark 1.1 (Evaluating the algebraic error estimator). We refer to [45, Remark 4.3] [55, Section 4] for more details on how to evaluate the algebraic error estimator. In our case, we follow the same process described in the references above, i.e. in order to compute the algebraic error estimator η_{alg} at iteration i of the algebraic solve, we carry out a few j additional iterations. Thus, the degrees of freedom required for estimating the algebraic error at iteration i will be calculated by evaluating a difference between the degrees of freedom obtained after i and $i+j$ iterations respectively.

Before choosing the preconditioner, it is necessary to evaluate the algebraic error estimator. To do this, we need to have performed k_{rest} iterations with the linear solver. This way, if need be, the algebraic estimator will be evaluated for the k_{rest}-th iteration of the algebraic solve and we will run a series of k_{rest} additional iterations (that is to say, up to iteration number $2 * k_{\text{rest}}$).

The appropriate response to the third point raised here is not obvious. If we have to restart new iterations after having evaluated the algebraic estimator and decided on the preconditioner to be used and if we reuse the last intermediary solution $x_{k_{\text{rest}}}$ as a starting vector for the preconditioned solve, then we can say that the iterative solver is restarted. In the case of BiCGStab, in which only the initial guess is reinitialized while the other data, such as the arbitrary vector and the research direction are lost, the effect of such a restart remains a big question mark since it is missing in the literature. Moreover, in case a linear solver other than BiCGStab is used and within which the restart option is provided for, like GMRES for example, we see no reason why this option should not be applied in this context. Hence, we have decided not to reuse the intermediary solution $x_{k_{\text{rest}}}$ in BiCGStab, and reuse it in GMRES (Section 1.1.7).

Algorithm 8 summarizes the approach followed with the configurations outlined above and based on error estimators for solving nonlinear nonstationary PDEs stemming from simulations in geosciences with an adaptive choice of preconditioners.

Algorithm 8 Adaptive Choice of Preconditioner

Inputs: $\mathbf{M}_{\text{S}}$, $\mathbf{M}_{\text{W}}$, γ_{prec}, k_{rest}, $k_{\max}$, $t_{\max}$, $n_{\max}$
for $t = t_0$ **until** $t_{\max}$ %Time loop
  Discretize the system
  Let $n = 0$, and $\mathbf{M} = \mathbf{M}_{\text{W}}$
  while non-convergence and $(n < n_{\max})$ %Newton loop
    Linearize the system
    if $(t > t_0)$
      if $(n == 0)$
        Evaluate the a posteriori error estimators η_{alg} and η_{sp}
        if $(\eta_{\text{alg}} < \gamma_{\text{prec}} * \eta_{\text{sp}})$
          Choose the less robust preconditioner $\mathbf{M} = \mathbf{M}_{\text{W}}$
        else
          Choose the more robust preconditioner $\mathbf{M} = \mathbf{M}_{\text{S}}$
        end if
      end if
    end if
    Solve the linear system with preconditioner $\mathbf{M}$, up to $k_{\max}$ iterations maximum
    $n = n + 1$
  end while
end for

We should note that it is possible to combine the adaptive approach of preconditioner choice with stopping criteria. In this second version of the adaptive preconditioner's choice algorithm, the a posteriori algebraic error estimator η_{alg} has to be evaluated for every linear system. The corresponding process is described below.

For every linear system of the simulation, we compute η_{alg}. Then if the ratio $\frac{\eta_{\text{alg}}}{\eta_{\text{sp}}}$ is smaller than γ_{alg}, this system is considered to have converged. Otherwise, we choose a preconditioner in the same way as in the first version (Algorithm 8). This helps to resolve the issue relating to the interruption of simulations. However, the resulting solution is not always satisfactory, and leads to an increase in the number of Newton iterations performed. Therefore, in order to reduce the negative effect of these a posteriori stopping criteria on Newton iterations, the following configuration has to be applied: in case the solution of the system is stopped by the a posteriori stopping criterion, then all the subsequent intermediary series of iterations necessary for evaluating the algebraic estimators η_{alg} of the same time step are performed with the robust preconditioner $\mathbf{M}_{\text{S}}$. Algorithm 9 describes the algorithm of the adaptive choice of preconditioners with stopping criteria based on a posteriori error estimators.

Algorithm 9 Adaptive Choice of Preconditioner with Stopping Criteria

Inputs: $\mathbf{M}_{\text{S}}$, $\mathbf{M}_{\text{W}}$, γ_{prec}, γ_{alg}, k_{rest}, $k_{\max}$, $t_{\max}$, $n_{\max}$

1: **for** $t = t_0$ **until** $t_{\max}$ %Time loop
2: Discretize the system
3: Let $n = 0$, $\mathbf{M} = \mathbf{M}_{\text{W}}$, $StCr = 0$
4: **while** non-convergence and $(n < n_{\max})$ %Newton loop
5: Linearize the system
6: **if** $(t > t_0)$
7: **if** $(StCr == 1)$ $\mathbf{M} = \mathbf{M}_{\text{S}}$ **else** $\mathbf{M} = \mathbf{M}_{\text{W}}$ **end if**
8: Evaluate the a posteriori error estimators η_{alg} and η_{sp}
9: **if** $(\eta_{\text{alg}} < \gamma_{\text{alg}} * \eta_{\text{sp}})$
10: $n = n + 1$, $StCr = 1$
11: Go to line 5
12: **end if**
13: **if** $(\eta_{\text{alg}} < \gamma_{\text{prec}} * \eta_{\text{sp}})$
14: Choose the less robust preconditioner $\mathbf{M} = \mathbf{M}_{\text{W}}$
15: **else**
16: Choose the more robust preconditioner $\mathbf{M} = \mathbf{M}_{\text{S}}$
17: **end if**
18: **end if**
19: Solve the linear system with preconditioner $\mathbf{M}$, up to $k_{\max}$ iterations maximum
20: $n = n + 1$
21: **end while**
22: **end for**

1.1.3 Runtime platform

In this section focused on the study of the adaptive approach of global preconditioning, we run numerical tests on the following execution platform: a Linux cluster that delivers a peak performance of 110 TeraFlops, and consists of dual-socket 212 nodes with shared memory, with a memory capacity of 32 gigabytes and eight-core Sandy Bridge processor having a clock rate of 2.60 GHz (Intel Xeon E5-2670). We allocate one compute node and run parallel computations on

16 processors.

1.1.4 Computing framework

We use real data sets taken from reservoir simulations run on IFPEN's prototype for reservoir simulation. We use the ALIEN interface which provides a modern, uniform and generic API for a wide range of linear solver libraries including Petsc [16], IFPSolver [64] and MCGSolver * [7, 8]. These last two solvers are the ones we are going to use for the numerical tests presented in this chapter. These solvers have been developped at IFPEN and provide linear solver algorithms for IFPEN's industrial simulations, such as Krylov solvers BiCGStab and GMRES and preconditioners, like ILU(0) and CPR-AMG.

1.1.5 Presentation of the study cases

SPE10 is an immiscible two-phase reservoir model (oil + water) with a strong heterogeneity of petrophysical data that was introduced during the "SPE10th Comparison Solution Project" to compare different upscaling methods [38]. It is composed of 60 vertices in X axis, 220 vertices in Y axis and 85 vertices in Z axis. This corresponds to a total of over a million vertices. The model contains five wells: four producing wells located at the ends of the grid and an injection well in the center. This case is also known as a case in which we are likely to encounter difficulties concerning the algebraic solving with the linear solver.

3DBlackOil is a test case that corresponds to the simulation of the Black-Oil model. This model consists of three phases which are water, oil and gas. The oil phase comprises two types of components: non-volatile oil and volatile oil which can be referred to as oil component and gas component respectively. The reservoir dealt with in this test case is a three-dimensional domain (4750m $\times$ 3000m $\times$ 114m) discretized with a CPG (Corner Point Geometry) grid. We consider an anisotropic heterogeneous reservoir with a porosity of 30%, three horizontal and two vertical permeability layers. We consider a gas injection in the initially unsaturated reservoir. A gas-injection well punctures a corner of the reservoir in the vertical direction (Z axis) and a producing well is located in the opposite corner. The domain is divided into a grid $38 \times 24 \times 5$ cells. The simulation lasts either 30 days or 2000 days. The data, the constraints and the PVT properties (pressure-volume-temperature) are extracted from the SPE1 model designed to simulate a three-dimensional Black-Oil reservoir, see [110].

1.1.6 Numerical results and comments

Table 1.1 shows the results obtained with a fixed preconditioner for the entire simulation, with three test cases: SPE10-30days, SPE10-2000days and 3DBlackOil. Table 1.2 provides the results obtained with the adaptive preconditioning of Algorithm 8 on BiCGStab with the same test cases. Here, for every system $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$, the algebraic estimate η_{alg} is evaluated at the k_{rest}-th iteration of BiCGStab preconditioned by ILU(0). According to the parameter γ_{prec} and to Equation (1.2), the preconditioner is chosen as detailed earlier. Then, the preconditioned linear solver with this ad

*. MultiCore, ManyCore, MultiGPU Solver

hoc preconditioner is used for solving the system, and can run up to $k_{\max}$ iterations. On the other hand, the preconditioner choice is made only once per time step, and the same preconditioner is used for the other systems of the same time step (Algorithm 8).

Table 1.1 – *The solve time (SolvTim) and the total number of iterations (IT) needed for convergence of IFPSolver's BiCGStab for (SyS) number of systems, with the preconditioner set for the whole simulation.*

Test case	Preconditioner	SyS	IT	SolvTim
SPE10-30days	ILU(0)	38	3815	163.84
	CPR-AMG	35	594	133.39
SPE10-2000days	ILU(0)	141	44699	1983.9
	CPR-AMG	139	5352	1009.1
3DBlackOil	ILU(0)	334	20566	84.26
	CPR-AMG	342	1504	77.66

Table 1.2 – *The solve time (SolvTim) and the total number of iterations (IT) needed for convergence of IFPSolver's BiCGStab for (SyS) number of systems, where BiCGStab is restarted twice after $k_{\text{rest}} = 15$ iterations. The first two series of k_{rest} iterations are used to calculate the a posteriori estimates necessary for the adaptive choice of preconditioners for $\gamma_{\text{prec}} = 10^{-2}$.*

Test case	Adaptive ILU(0)/CPR-AMG				
	SyS	IT	SolvTim	Nb ILU(0)	Nb CPR-AMG
SPE10-30days	54	668	121.1	13	41
SPE10-2000days	335	10444	1251.3	59	276
3DBlackOil	345	1641	78.52	7	338

The parameter k_{rest} is fixed at 15 in order to leave a reasonable minimum of iterations to make a judicious choice of preconditioner. We effectively get some improvement in computing time when using simple adaptive preconditioning (Algorithm 8) in comparison with when using only the preconditioner $\mathbf{M}_{\text{W}}$ for the entire simulation. In some cases, the adaptive algorithm yields less computing time than even the preconditioner $\mathbf{M}_{\text{S}}$. Since we don't know in advance what the better choice is, the adaptive algorithm guides the iterative solver towards the adequate preconditioner.

1.1.7 Adaptive choice with restarted GMRES

As shown in the numerical results cited above, the adaptive preconditioner's choice algorithm applied to BiCGStab iterative solver sometimes delivers small gains in number of iterations or in computing time. Therefore, trying another iterative solver may be a possible solution for testing the performances of the adaptive preconditioner's choice algorithm we are using.

GMRES is one of the most popular methods for the solution of linear systems with non-symmetric matrices. It is said to be optimal as the approximate solution is constructed in the Krylov sub-space that minimizes the euclidean norm of the residual [125]. The restart of GMRES is recommended when the highest attainable precision is not satisfactory, such as in the case of rounding errors, and when it becomes necessary to stop the iterations. It is also recommended for saving the memory

space when the size of Krylov subspace becomes too big to store all the basis vectors. Thus the restart allows to limit the memory resources necessary for the computing of the solution. However, the restart may compromise the global optimality. In fact, without restart GMRES converges exactly in at most N iterations if N is the size of the system while a restarted GMRES may not converge: There may be a noticeable plateau effect at every restart.

In the remainder of this section, we focus on the GMRES method in its restarted version GMRES(m) [125] as the iterative linear solver for the adaptive algorithm of preconditioner's choice based on a posteriori error estimators. In this case, the method is restarted when the dimension of the Krylov subspace reaches m, and the current approximate solution computed at each cycle becomes the initial guess of the coming cycle. The matter of the choice of the restart parameter is not to be taken lightly as it can remarkably affect the convergence [81, 52]. In the configuration of GMRES solver for the adaptive preconditioners' choice algorithm, the parameter m is appropriately set so as to satisfy the following two main criteria :

— m should be relatively small compared to n (the matrix size) to keep reasonable computing and memory needs.

— m should not be too small, because in this case the computed a posteriori error estimators would not be very coherent. The reason for this is that the algebraic error estimators are computed from approximate solutions stemming from two successive cycles: it would make no sense to compute those quantities after one cycle of only 5 iterations for example for GMRES.

1.1.7.1 Restart configuration for GMRES

As a first step, it is necessary to decide on the dimension m of the Krylov subspace in which the solutions of the linear systems will be computed. A quick search in the literature gives us an empirical order of magnitude of the dimensions taken in practice for computings done in different fields not necessarily related to the scope of reservoir simulation. Therefore, we had to check the convergence of the restarted GMRES solver after cycles of m iterations for all the linear systems in a simulation, at least with the most robust preconditioner. After experimenting with different values of m, we have opted for the smallest tested value with which the convergence has been achieved: $m = 100$.

After that, some GMRES configurations have been experimented for the adaptive preconditioners' choice algorithm. The first configuration referred to as (V1) is summarized as follows. For computing the algebraic error estimates, two consecutive calls to GMRES(m) solver are made (see Remark 1.1). The first call is made starting from a zero value initial guess $\mathbf{x}_0$, and a first intermediary solution $\mathbf{x}_1$ is retrieved after a cycle of m iterations. A second call to the solver is subsequently made starting from solution $\mathbf{x}_1$, and sends back a second intermediary solution $\mathbf{x}_2$ after a cycle of m iterations. Once the algebraic error estimator computed from $\mathbf{x}_1$ and $\mathbf{x}_2$, the preconditioner is chosen as previously described in Section 1.1.1, and the actual solving of the systems is carried out with the preconditioned GMRES(m), restarted as many times as necessary provided that the maximum total number of iterations is not exceeded, which is 5000 iterations or 50 restart cycles (each cycle being equivalent to 100 iterations).

With this first configuration, the adaptive preconditioners' choice algorithm takes the form

Algorithm 10 Adaptive Choice of Preconditioner with GMRES (V1/ V2 Versions)

Inputs: $\mathbf{M}_S$, $\mathbf{M}_W$, γ_{prec}, k_{rest}, k_{max}, t_{max}, n_{max}, m
for $t = t_0$ **until** t_{max} %Time loop
  Discretize the system
  Set $n = 0$, and $\mathbf{M} = \mathbf{M}_W$
  while non-convergence and $(n < n_{max})$ %Newton loop
    Linearize the system
    if $(t > t_0)$
      if $(n == 0)$
        Compute a 1st intermediary solution after one cycle of GMRES:
          $\mathbf{x}_1 = \text{GMRES}(m)(\mathbf{A}, \mathbf{b}, \mathbf{x}_0 = \mathbf{0})$ / $\mathbf{x}_1 = \text{GMRES}(m/2)(\mathbf{A}, \mathbf{b}, \mathbf{x}_0 = \mathbf{0})$
        Compute a 2nd intermediary solution after another cycle of GMRES:
          $\mathbf{x}_2 = \text{GMRES}(m)(\mathbf{A}, \mathbf{b}, \mathbf{x}_0 = \mathbf{x}_1)$ / $\mathbf{x}_2 = \text{GMRES}(m/2)(\mathbf{A}, \mathbf{b}, \mathbf{x}_0 = \mathbf{x}_1)$
        Evaluate the a posteriori error estimators η_{sp} and η_{alg} from $\mathbf{x}_1$ and $\mathbf{x}_2$
        if $(\eta_{alg} < \gamma_{prec} * \eta_{sp})$
          Choose the less robust preconditioner $\mathbf{M} = \mathbf{M}_W$
        else
          Choose the more robust preconditioner $\mathbf{M} = \mathbf{M}_S$
        end if
      end if
    end if
    Compute the final solution with GMRES preconditioned by $\mathbf{M}$:
      $\mathbf{x} = \text{GMRES}(m)(\mathbf{M}, \mathbf{A}, \mathbf{b}, \mathbf{x}_0 = \mathbf{x}_2)$
    $n = n + 1$
  end while
end for

described in Algorithm 10 (the intermediary solve steps to consider are colored in red).

It will be noted that the computing stage of the algebraic error estimator will have taken two GMRES cycles, that is to say 200 iterations, which is significant in comparison with what we got with BiCGStab, for a pretreatment phase that logically should not represent a very considerable computational load. Thereafter, in an attempt to reduce this computational load, we have considered a second configuration (V2) such as the first two solver calls necessary for the evaluation of the algebraic error estimator are made with a Krylov basis that is reduced by half GMRES($m/2$), while the other calls for the actual solving of the system are made with a basis whose size is m : GMRES(m). This approach reflects that the first two calls to GMRES($m/2$) represent merely a pretreatment step that will give only intermediary solutions and is therefore not worth spending as many iterations as the rest of the calls to GMRES(m) that come after and yield the final solution of the system. With this second configuration, the adaptive preconditioner's choice algorithm corresponds to Algorithm 10 with the blue intermediary solve steps.

The other possibility we consider worth exploring is what could happen if the solver was not restarted after the evaluation phase of the algebraic error estimator. For this reason, we have thought of a third version which is in line with configuration V1, in the sense that the actual solving of the system is done with a basis of m vectors, and that the first intermediary solution $\mathbf{x}_1$ is computed after a first cycle with a basis of m vectors. Yet, the difference is that for computing the second intermediary solution, we do not start over a new cycle with a basis of m vectors constructed from $\mathbf{x}_1$, but we continue constructing the Krylov basis of $2m$ vectors where the solution $\mathbf{x}_2$ is to be sought. In other words, $\mathbf{x}_2$ would be the solution computed after a cycle of $2m$

iterations whereas $\mathbf{x}_1$ is computed just with the first half of that cycle (or the first m vectors of the constructed basis).

Algorithm 11 is formed by the classical stages of restarted GMRES, but it also includes the evaluation stage of error estimates and that of changing the preconditioner.

Algorithm 11 Adaptive Choice of Preconditioner with GMRES (V3 Version)

Inputs: Initial guess $\mathbf{x}_0$, Krylov subspace dimension m, number es of iterations before evaluating the algebraic error estimator, $(m+1) \times m$ Hessenberg matrix $\overline{\mathbf{H}}_m$ initialized to $\mathbf{0}$, initial preconditioner $\mathbf{M}$.

1: $InitPrec = true$
2: Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A} \cdot \mathbf{x}_0$, $\beta = ||\mathbf{r}_0||_2$ and $\mathbf{v}_1 = \beta^{-1}\mathbf{r}_0$
3: $j := 1$
4: **while** (non-convergence and $j < m$)
5: Compute $\mathbf{z}_j := \mathbf{M}^{-1} \cdot \mathbf{v}_j$
6: Compute $\mathbf{w} := \mathbf{A} \cdot \mathbf{z}_j$
7: **for** $i = 1 \ldots j$ **Do**
8: $\mathbf{H}_{i,j} := (\mathbf{w}, \mathbf{v}_i)$
9: $\mathbf{w} := \mathbf{w} - \mathbf{H}_{i,j}\mathbf{v}_i$
10: **end for**
11: $\mathbf{H}_{j+1,j} = ||\mathbf{w}||_2$
12: **if** ($\mathbf{H}_{j+1,j} \neq 0$)
13: $\mathbf{v}_{j+1} := \mathbf{H}_{j+1,j}^{-1}\mathbf{w}$
14: **else**
15: Set $m = j$ and go to line 31
16: **end if**
17: // Construct an (intermediary) approximate solution:
18: **if** (InitPrec and j mod $es == 0$)
19: Compute $\mathbf{y}_j$ that minimizes $||\beta\mathbf{e}_1 - \overline{\mathbf{H}}_j \cdot \mathbf{y}||_2$
20: Save the value $\mathbf{x}_j = \mathbf{x}_0 + \mathbf{Z}_j \cdot \mathbf{y}_j$ where $\mathbf{Z}_j = [\mathbf{z}_1, \ldots, \mathbf{z}_j]$
21: **end if**
22: // Evaluate the algebraic estimator:
23: **if** (InitPrec and $j == 2 * es$)
24: Compute η_{alg} from $\mathbf{x}_j$ and $\mathbf{x}_{es}$. // See Remark 1.1 for evaluating the algebraic error estimator
25: Set the preconditioner $\mathbf{M}$ according to the value of $\frac{\eta_{\text{alg}}}{\eta_{\text{sp}}}$
26: $InitPrec = false$
27: **end if**
28: $j := j + 1$
29: **end while**
30: Compute $\mathbf{y}_m$ that minimizes $||\beta\mathbf{e}_1 - \overline{\mathbf{H}}_m \cdot \mathbf{y}||_2$, with $\mathbf{e}_1 = (1, 0, \ldots, 0)^{\mathrm{T}}$
31: Update the initial guess $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{V}_m \cdot \mathbf{y}_m$ where $\mathbf{V}_m = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$
32: // Restart: **if** (convergence) stop **else** update $\mathbf{x}_0 := \mathbf{x}_m$ and go to line 2 **end if**

1.1.7.2 Numerical results and comments

We compare the performance of the three preceding configurations on the test cases (SPE10-30days/3DBlackOil) to determine the one that we will keep for subsequent simulations. Table 1.3 includes for the three configurations the solve times, the total number of iterations and the number of times each of the two preconditioners is used. The number of times the preconditioner $\mathbf{M}_{\mathrm{W}}$ is used shows clearly that configurations V2 and V3 are less efficient. They take significantly more

∗. Each GMRES cycle is equivalent to 100 iterations.

Table 1.3 – *The number of times $\mathbf{M}_S = ILU(0)$ and $\mathbf{M}_W = Poly$ were used, the solve time (SolvTim), the total number of iterations (IT) during the 3DBlackOil simulation with the 3 configurations V1, V2 (Algorithm 10) and V3 (Algorithm 11) of adaptive preconditioner choice for MCGSolver's GMRES, $\gamma_{\text{prec}} = 10^{-2}$.*

Test case	Configuration	Nb of switches to $\mathbf{M}_W$	Nb of switches to $\mathbf{M}_S$	IT*	SolvTim
SPE10-30days	V1	34	27	89 (*100)	606.69
	V2	3	57	$\approx$ 121 (*100)	652.08
	V3	13	36	$\approx$ 157 (*100)	1114.07
3DBlackOil	V1	25	472	1494 (*100)	764.50
	V2	6	412	$\approx$ 1779 (*100)	927.19
	V3	6	411	$\approx$ 1857 (*100)	1031.82

iterations and solve time than configuration V1. This can be due to the fact that in these two configurations, the algebraic error estimators we encounter are rather significant, which shows a large difference between the two intermediary solutions computed $\mathbf{x}_1$ and $\mathbf{x}_2$.

In fact, in configuration V2 the two solutions are constructed after half cycles of $m/2$ iterations, which are, as it has been observed earlier, insufficient to solve all the systems of the simulation. Therefore, the high values found in the a posteriori algebraic error estimator can be explained by the poor quality of the intermediary solutions.

Whereas in configuration V3, the first intermediary solution $\mathbf{x}_1$ is computed after one cycle of m iterations, and greatly differs from the second intermediary solution $\mathbf{x}_2$. Indeed, these two solutions are not constructed within bases of the same dimension. The one in which $\mathbf{x}_2$ is constructed contains that where $\mathbf{x}_1$ was built, but contains twice as many vectors, which leads to the wide gap between the two entities. Not surprisingly in the end the total number of iteration and the solve time are highest with this configuration V3 since we are using bases of $2m$ elements to compute the second intermediary solution $\mathbf{x}_2$.

For all these reasons, we have decided to keep only the first configuration V1 for which we present the numerical results on the test cases SPE10-30days and 3DBlackOil in Table 1.4.
We can thus compare the preconditioners $\mathbf{M}_W = Poly$ and $\mathbf{M}_S = ILU(0)$ (ResOpt = 0) with the adaptive preconditioner (ResOpt = 1). The third and fourth columns of that table show the number of systems preconditioned by $\mathbf{M}_W$ and $\mathbf{M}_S$ respectively.

The first conclusion from analyzing this table is that we obtain a gain with the adaptive strategy of global preconditioning on some test cases we have carried out. Therefore, the observations made in Section 1.1.6 for BiCGStab remain valid for GMRES as well. This means that changing the solver does not substantially affect the performance of the adaptive algorithm. Yet, this method remains interesting as it allows to steer the choice of the preconditioner during the simulation.

∗. GMRES did not converge with cycles of 50 iterations for this test case.

Table 1.4 – *The solve time (SolvTim) and the total number of iterations (IT) needed for convergence of MCGSolver's GMRES for (SyS) number of systems, where we fixed the value $\gamma_{\text{prec}} = 10^{-2}$, and GMRES is restarted:*
- as much as needed until the convergence is reached (ResOpt = 0),
- 2 times first for the sake of a posteriori estimates computation then as much as needed until the convergence is reached (ResOpt = 1).

Preconditioner			**Nb of Calls to Poly**	**Nb of Calls to ILU(0)**	**Total**	
Test case	KrylovDim	ResOpt	SyS	SyS	IT	SolvTim
SPE10-30days	50	0	38	-	403 (*50)	1800.64
			-	39	230 (*50)	583.24
		1	24	59	203 (*50)	535.03
	100	0	37	-	150 (*100)	1427.05
			-	39	90 (*100)	506.66
		1	34	27	89 (*100)	606.69
3DBlackOil	100*	0	338	-	1604 (*100)	1474.46
			-	336	1495 (*100)	758.81
		1	25	472	1494 (*100)	764.50

1.2 Local adaptive preconditioning based on a posteriori error estimates

The global approach of adaptive preconditioning based on a posteriori error estimates for dynamic simulations in Geosciences has experimentally proven to be satisfactory, yet the results of applications on some real test cases show that the obtained gains remain limited and still below what had been aspired for. The next step goes further towards the exploitation of a posteriori error estimates in order to evaluate other adaptive preconditioning strategies. Henceforth, we will not only rely on criteria on the global sum of estimates on all mesh elements, but we will, this time, take into account the local values of estimates on every mesh element so as to choose the preconditioning methods.

Following the same rationale of adaptive mesh refinement [141], in which we refine the mesh where the values of error estimates are the highest in order to get significantly more precise results for those areas, the idea we develop in this chapter is that for every linear system to be solved, we want to construct block preconditioners in which we make a distinction between the blocks of the matrix that correspond to mesh elements with the highest error estimates and the rest of the blocks that would correspond to elements with insignificant or non-existent error. In fact, for the former blocks, we opt for more robust preconditioners even if they are more computationally expensive, while for the latter ones we can settle for less robust preconditioners that are therefore faster. Here, however, the expected result is to improve the computing performances. By targeting the parts of the matrix in which we know that algebraic errors exist, we improve the performance (iterations and/or computing time) by restricting as much as possible the use of robust preconditioners to the areas with the most significant errors. This idea is consistent with the concept of domain decomposition. One Domain Decomposition-based preconditioning strategy that we consider in the remainder of this chapter is the Block-Jacobi method.

The quality of Block-Jacobi preconditioners is strongly dependent on the coupling between

unknowns related to distinct local diagonal blocks. The number of non-zero elements in the off-diagonal blocks with respect to the number of non-zero elements in the diagonal blocks of **A** can give an indication of the coupling strength. In case of a strong coupling, Block-Jacobi preconditioners are generally inadequate and should be avoided. For loosely-coupled subdomains, Block-Jacobi preconditioners can be reasonably effective. In our study cases, the matrices obtained from the simulator fall under this later category as the non-zero entries are mainly located in the diagonal blocks, and are dominant in absolute value over the few non-zero entries located in the off-diagonal blocks. Furthermore, it is possible to improve a Block-Jacobi preconditioner by increasing the size of the diagonal blocks, for example by following the strategy suggested in Section 1.2.3.2. This necessarily leads to a decrease in the number of diagonal blocks and thereby to a reduction in the degree of parallelism in generating and applying the preconditioner.

Taking into account the characteristics of the matrices dealt with and the features of each of the methods discussed earlier in the state-of-the-art section, we decide to use a Block-Jacobi preconditioning method as it has a lower computational cost while more communication is necessary between subdomains for the other domain decomposition methods. For more details about these methods, we refer the reader to the article [28].

1.2.1 Error distribution and matrix conditioning

We investigate in this section if there is a link between a posteriori error estimates and the conditioning of the matrix. To elaborate on this approach from a linear algebra point of view, especially how the matrices and the vectors on which we are working are built, let us consider first a simple structured 2D mesh composed of four vertices in each direction. Next, it is assumed that the nodes are ordered line by line, from bottom to top and from left to right (Figure **1.1**).

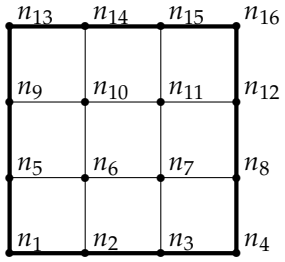


Figure 1.1 – *Example of a simple* 4×4 *mesh*

If we solve a PDE model with two unknowns: $u = \begin{pmatrix} u^{(1)} \\ u^{(2)} \end{pmatrix}$, then the assembly of the global system at every Newton iteration is done by blocks of two entries. Let $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ be the linear system produced by the discretization of the PDE with a two-point finite volume scheme, then we have:

• **x** is a vector of length 32 (2 unknowns, 16 nodes) that contains the discrete values of the solution of the PDE on all the mesh elements. Thus, the first element of the vector **x** represents the value of the unknown $\mathbf{u}_1$ on the first mesh element. The second element of the vector **x** represents the value of the unknown $\mathbf{u}_2$ on the first mesh element and so on until the last element of the vector for the last mesh element according to the prescribed order of the mesh elements.

• **b** is a vector of length 32 that contains the discrete values of the right hand side function of the PDE on all the mesh elements.

• **A** is a 32×32 matrix with coefficients of the equations that link the unknowns of the solution over the mesh. For example, every row i of the matrix contains the coefficients associated to the i-th unknown contained in the solution vector **x** (these terms are located on the diagonal of the matrix), and the coefficients associated to the other unknowns of which the diagonal unknown depends (these are the off-diagonal entries of the matrix). Hence, if there are p non-zero off-diagonal elements on this line, this means that the i-th unknown depends only on p other unknowns.

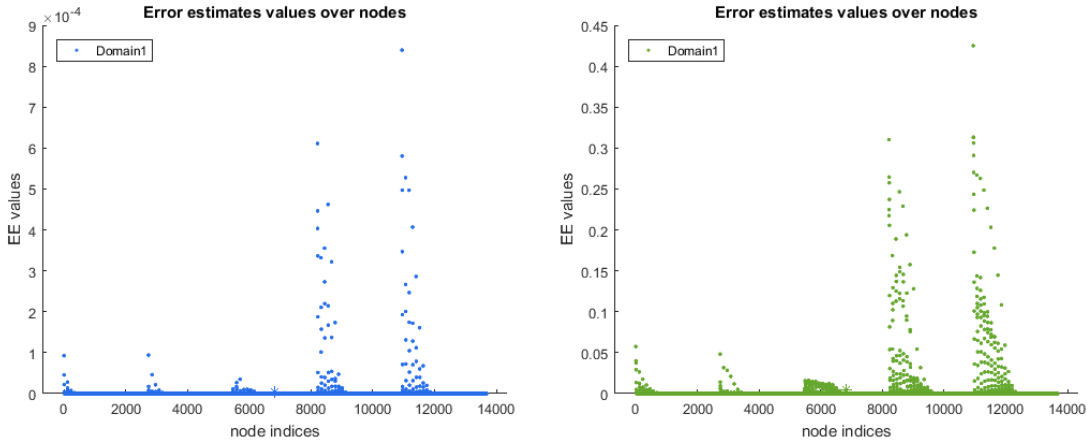
Thus, as the shape of the matrix below shows (Figure 1.2), we can distinguish on the one hand the diagonal blocks $(\mathbf{A}_{ii})_{1\leq i\leq 16}$ of the matrix that are associated to the meshes because they contain the dependencies between the unknowns of the solution $u^{(1)}$ and $u^{(2)}$ on the same mesh element, and on the other hand the off-diagonal blocks $(\mathbf{A}_{ij})_{i\neq j}$ of the matrix that contain the flux terms and represent for the i-th element the solution's dependencies on its values on the j-th mesh element.

The diagonal blocks are assembled through the mesh elements while the off-diagonal blocks are assembled through the links (or interface between two mesh elements on which the flows are discretized).

$$\mathbf{A} = \begin{bmatrix}
\mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{1,5} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \mathbf{A}_{2,3} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{2,6} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{A}_{3,2} & \mathbf{A}_{3,3} & \mathbf{A}_{3,4} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{3,7} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{A}_{4,3} & \mathbf{A}_{4,4} & \mathbf{A}_{4,5} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{4,8} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{5,4} & \mathbf{A}_{5,5} & \mathbf{A}_{5,6} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{5,9} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{A}_{6,1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{6,5} & \mathbf{A}_{6,6} & \mathbf{A}_{6,7} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{6,10} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{A}_{7,2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{7,6} & \mathbf{A}_{7,7} & \mathbf{A}_{7,8} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{7,11} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{A}_{8,3} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{8,7} & \mathbf{A}_{8,8} & \mathbf{A}_{8,9} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{8,12} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{9,4} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{9,8} & \mathbf{A}_{9,9} & \mathbf{A}_{9,10} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{9,13} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{10,5} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{10,9} & \mathbf{A}_{10,10} & \mathbf{A}_{10,11} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{10,14} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{11,6} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{11,10} & \mathbf{A}_{11,11} & \mathbf{A}_{11,12} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{11,15} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{12,7} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{12,11} & \mathbf{A}_{12,12} & \mathbf{A}_{12,13} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{12,16} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{13,8} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{13,12} & \mathbf{A}_{13,13} & \mathbf{A}_{13,14} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{14,9} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{14,13} & \mathbf{A}_{14,14} & \mathbf{A}_{14,15} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{15,10} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{15,14} & \mathbf{A}_{15,15} & \mathbf{A}_{15,16} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{16,11} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{16,15} & \mathbf{A}_{16,16}
\end{bmatrix}$$

Figure 1.2 – *Structure of matrix* **A**

• The numerical analysis of a posteriori error estimates returns a vector **EE**, of the same length as the previous vectors **x** and **b**, that contains the estimation of the algebraic error on the approximate solution for every mesh element. Like the matrix **A**, the vector **EE** is sparse: It contains far more zero elements than non-zero elements as demonstrated in Figures 1.3 and 1.4 which display the values of error estimates over nodes for the test cases studied. This enables us to focus on the location of non-zeros in the error estimates vector **EE** to ensure that we have more accurate computations for these areas of the mesh. One approach of doing this consists in applying locally a more robust preconditioning. But before that, we would like to make sure that the values of the algebraic error estimates are correlated with the local conditioning of the submatrices comprised in the block-diagonal. Also, it should be mentioned that if the value of the error estimate is large on the i-th mesh element, it would be of interest to check the conditioning of the diagonal block $\mathbf{A}_{ii}$ because it is this submatrix that contains the local dependencies between the solution components



(a) Mid-simulation at time step : $t = t_{42}$ *(b) End of simulation at time step : $t = t_{95}$*

Figure 1.3 – *Distribution of algebraic EE over main domain during simulation (3DBlackOil test case)*

on this element.

Therefore, we investigate experimentally if there is a link between the values of error estimates and the conditioning of the discretization matrix using data from real test cases. We consider two test cases presented earlier in this chapter (3DBlackOil, SPE10Layer85) for which we extract the discretization matrices and a posteriori error estimates during specific time steps from the middle and from the end of the simulation, where the error estimates vector has the most of non-zero elements. Then we consider mesh partitions of a given size (*size*), without permutation, that is to say by keeping the initial order of elements. This way, we have retrieved $(\mathbf{EE}_i)_i$ subvectors of $\mathbf{EE}$ that we map with $\mathbf{A}_{ii}$ the corresponding diagonal submatrices of $\mathbf{A}$. After that, we compute on one hand $\left(\overline{\mathbf{EE}_i}\right)_i$ the means of vector elements of $\mathbf{EE}_i$ subvectors and on the other hand the condition numbers of the submatrices $\mathbf{A}_{ii}$ and we compare these quantities. For information, the meshes are made of 4560 nodes for 3DBlackOil, 12997 nodes for SPE10Layer85. The number of unknowns per node is 3 for 3DBlackOil, and 2 for SPE10Layer85, which means that the global sizes of the matrices/vectors are 13680 for 3DBlackOil and 25994 for SPE10Layer85. Figures **1.5**, **1.6** represent the condition numbers with respect to the averages of error estimates locally evaluated by subdomain. We wish to clarify here that since we plot according to the common logarithm, we have represented zero averages by $\log_{10}(\overline{\mathbf{EE}_i}) = -40$ instead of $-\infty$ on the X-axis.

At first sight, we can distinguish in Figures **1.5** and **1.6** three clusters or clouds of points. First, we have a set of points at top right that contains subdomains in which the average of error estimates is non-zero ($\overline{\mathbf{EE}_i} \geq 10^{-8}$) and where the submatrices are relatively ill-conditioned ($\mathcal{K}(\mathbf{A}_{ii}) \geq 10^6$). Then there is a second set of points at bottom left that corresponds to the subdomains where error estimates are equal to zero ($\overline{\mathbf{EE}_i} == 0$) and where the submatrices are relatively well conditioned ($\mathcal{K}(\mathbf{A}_{ii}) \leq 10^6$). There is also a third set of points at the top left that consists of subdomains where the conditioning of the submatrices is rather high even if the error estimates are equal to zero.

Another point to outline is that while zooming the three clouds of points, we observe that the third cluster is the one that has the fewest points. This trend is being accentuated as we increase the size of partitions: The third cluster represents only 17% of the total number of points when

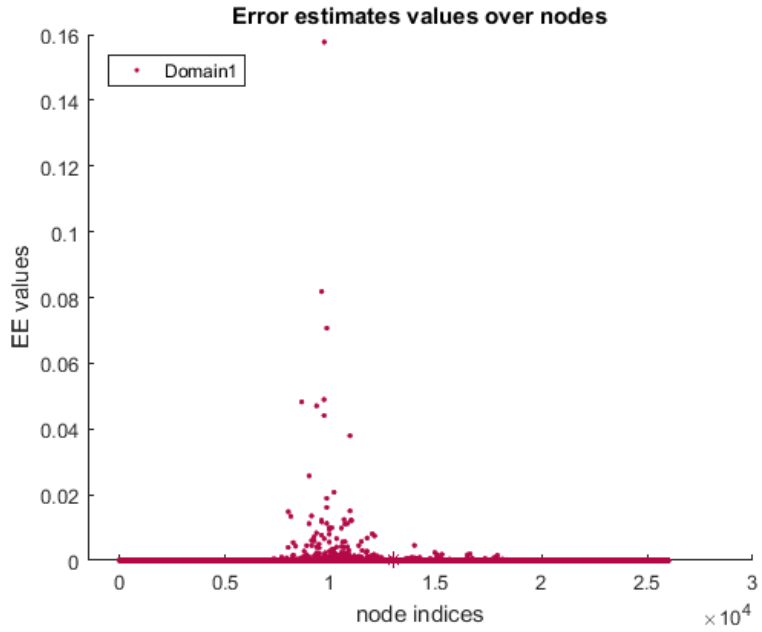
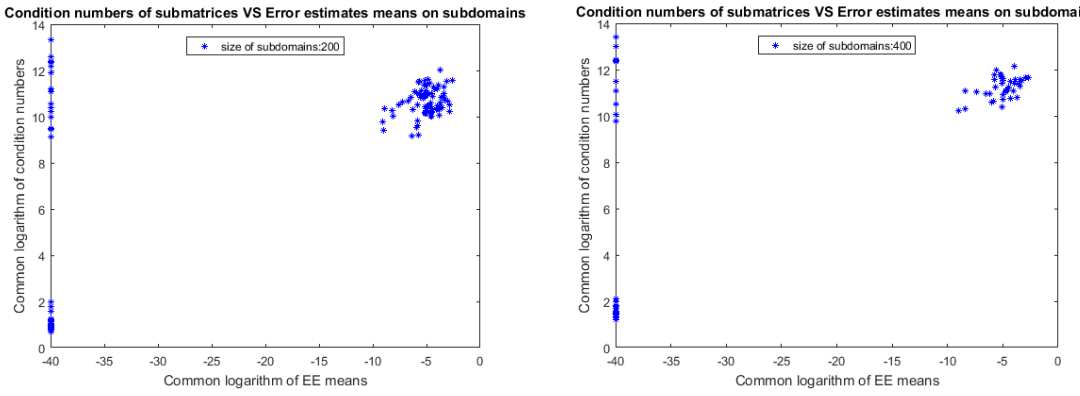
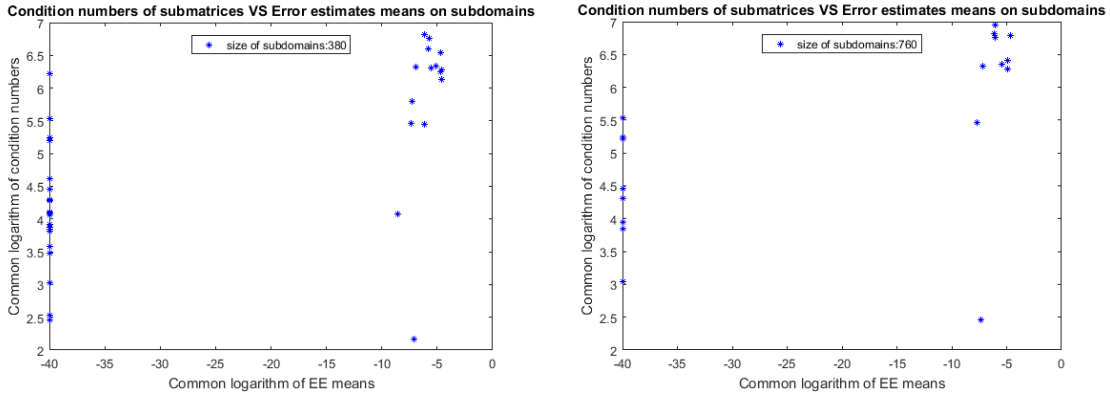


Figure 1.4 – *Distribution of algebraic EE over main domain at the end of simulation $t = t_{38}$ (SPE10Layer85 test case)*



(a) size $= 200$ *(b) size* $= 400$

Figure 1.5 – *Condition numbers of submatrices VS Error estimates means on subdomains for* 2 *different sizes of partitions at end of simulation $t = t_{38}$ (SPE10Layer85 test case)*



(a) size $= 380$ *(b) size* $= 760$

Figure 1.6 – *Condition numbers of submatrices VS Error estimates means on subdomains for* 2 *different sizes of partitions at mid-simulation* $t = t_{42}$ *(3DBlackOil test case)*

$size = 400$ for the test case SPE10Layer85 (Figure 1.5), while it does not include any point when $size = 760$ for the test case 3DBlackOil (Figure 1.6).
In short, the results above show that, experimentally, there is a correlation between error estimates and the conditioning of the subdomain matrices. The higher error estimates are on a set of mesh elements, the more badly conditioned the corresponding block in the discretization matrix will be. This being said, there may be some minor exceptions, mainly towards the end of the simulations where there are patterns that are difficult to predict (3DBlackOil test case).

1.2.2 Partitioning and permuting the matrix

The methodology used so far to partition the domain is natural and rather basic. It is similar to a 1D method [23] which consists in attributing $(N/\#Subdomains)$ consecutive rows (or columns) to each part, where N refers to the total number of rows and $\#Subdomains$ the number of parts. On the other hand, it is often possible to reduce intra-processor communication by partitioning the rows in a better way, not necessarily in a contiguous way, by using graphs to model the links between mesh elements (that correspond to non-zero entries of the matrix). In the standard graph model, each row (or column) of the matrix corresponds to a vertex of the undirected graph G. Hence, by partitioning the vertices of G, we partition the rows (or columns) of the matrix. However, the standard graph model is limited to symmetric matrices and is not suited for non symmetric matrices. To accommodate that, some solutions have been proposed such as the bipartite graph model [75] where we have two distinct sets of vertices: one set to represent the rows and another one for the columns. The edges between the two sets correspond to the non-zero coefficients of the matrix.

From graph theory perspective, one characteristic or feature of interest for a splitting, is to minimize the number of edges that relate two different parts. Another characteristic we would like to satisfy is to minimize the partition cost function. More precisely, let the undirected graph G be formed by n vertices whose weights are $(w_i)_{i=1,\dots,n}$ and p is an integer such that $\forall \;\; 1 \leq i \leq n, \;\; 0 < w_i \leq p$. We will denote by $\mathbf{C} = (C_{i,j})_{i,j=1,\dots,n}$ the adjacency matrix of the graph G, i.e. its element $C_{i,j}$ is

one when there is an edge from vertex i to vertex j, and zero when there is no edge.
Let $k \in \mathbb{N}^*$, a k-way partition of G is a set of non empty mutually disjoint subsets, $\mathcal{P} = \{P^{(1)}, P^{(2)}, \ldots, P^{(k)}\}$ such that $\bigcup_{i=1}^{k} P^{(i)} = G$. The cost of partition $\mathcal{P}$ is then defined by,

$$cost = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \notin P_i}}^{n} C_{i,j} \quad , \tag{1.3}$$

with $P_i \in \mathcal{P}$ being the subset to which the node i belongs. Thus, the aim of the classic problem of graph partitioning is to find a k-way partitioning such that the cost of the partition is as low as possible. This amounts to minimizing the number of edges (or the sum of their weights in the case of weighted graphs) between different partitions.

Partitioning is achieved by evenly distributing the vertices between the parts while taking the context into account. In fact, in order to distribute computations and to balance the loads, a good splitting would be the one that partitions the domain into subdomains with the same sizes. Figure 1.7 represents the matrix considered in Section 1.2.1, and its graph representation. To simplify the cuts, we consider that all the edges have the same weights.

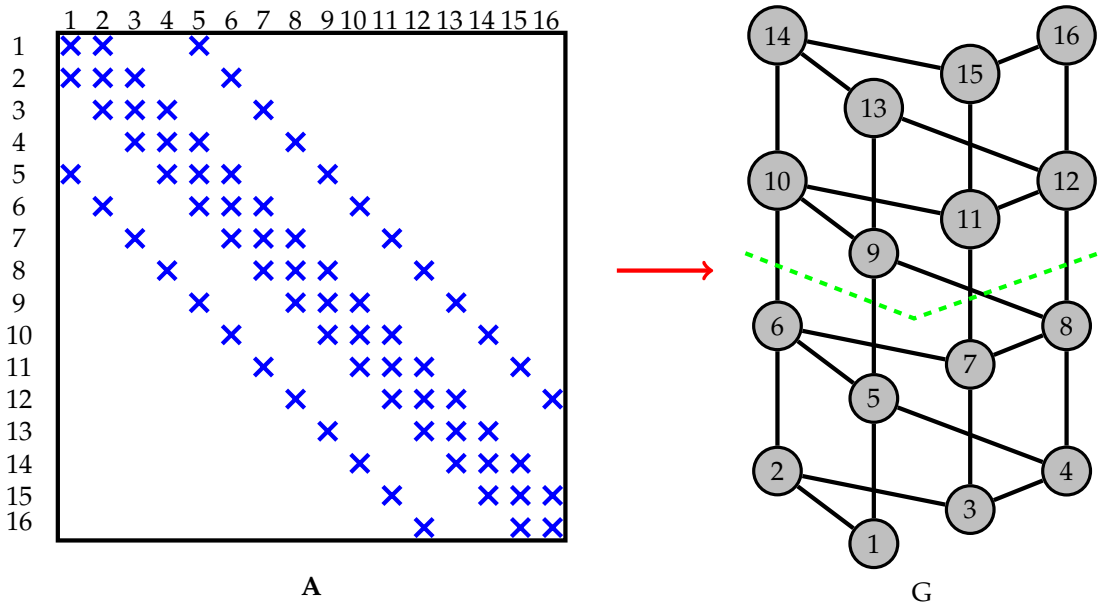


Figure 1.7 – *Constructing and partitioning the graph of the matrix into two parts.*

Since the problem of partitioning a graph into several subgraphs while minimizing the size of the cut is NP-hard, heuristics are generally used to get approximate solutions that are satisfactory enough to replace the optimal solution. There are various software programs that implement different heuristics. We select METIS as it is one of the most popular graph partitioners [83]. METIS is a library used to partition large meshes or irregular graphs. It can partition a non-structured graph into a number (k) of parts using either the multilevel recursive bisecting [85], or the multilevel k-way partitioning [84]. It is this second method that we have tested in this section.

As has been mentioned earlier, the standard graph model is limited to symmetric matrices while in our applications, the matrices are not necessarily so. Therefore, the partitioning is performed on the symmetric matrix $\mathbf{A} + \mathbf{A}^{\mathrm{T}}$ instead of $\mathbf{A}$.

To create an undirected graph from the symmetric matrix, we represent each of the rows (or the columns) by a vertex and each non-zero off-diagonal entry $\mathbf{A}_{ij}$ of the matrix $\mathbf{A}$ by an edge between vertices i and j. Then, we need to determine the weights of the edges and vertices. The weights of the vertices represent their importance or their complexity. In fact, it is possible to assign different weights to the vertices to reflect a computing charge that is more or less high at those locations. This helps to balance the size of partitions not in terms of number of vertices, but according to a sharper metric that reflects the complexity of each part. On the other hand, the weights of the edges reflect the degree of connection between vertices. It is a determining factor that affects the way the partitioner is going to partition the graph. The greater the weight between the two vertices, the more the partitioner will try to keep them in the same domain.

1.2.2.1 Standard partitioning configuration

Here we conserve the default configuration, where all the vertices have the same weight since they represent the rows/columns of the matrix. Therefore, there is no particular distinction between these ones, whereas for weighing the edges, we make use of the coefficients of the matrix. In fact, the weight of the edge connecting the vertex i and the vertex j, for example will be equal to the coefficient of the matrix at the intersection of row i and of column j.

Once the graph has been created, and all the weights set, we use the "metismex" function of METIS for partitioning the graph into k parts where k is a parameter chosen by the user. Each part is defined by a set of vertex indices. Then, this partitioning of the graph is reflected in the system by permutations of elements. For the matrix, it is necessary to permute the rows and columns to form the matrix blocks that correspond to the partitions. This is done as follows,

$$\mathbf{A}_p = \mathbf{R}\mathbf{A}\mathbf{R}^{\mathrm{T}}. \tag{1.4}$$

For the vectors (solution and right hand side), the permutation is made on the rows in order to gather the components belonging to the same partition. The permutation operation for a vector $\mathbf{v}$ is then written this way:

$$\mathbf{v}_p = \mathbf{R} \cdot \mathbf{v}; \tag{1.5}$$

with $\mathbf{R}$ being the matrix obtained by assembling the restriction operators $\mathbf{R}_s$ specific to each subdomain s:

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_1 \\ \hline \mathbf{R}_2 \\ \hline \vdots \\ \hline \mathbf{R}_{k-1} \\ \hline \mathbf{R}_k \end{bmatrix}.$$

If we go back to the example from Figure 1.7, where we have two parts, the first formed by the nodes from 1 to 8, the second including the nodes from 9 to 16, then the matrix $\mathbf{R}$ would match, in this case, the identity matrix.

If we take another example with a 9×9 matrix (the one at the left of Figure 1.8) whose graph partitioning would give the subdomains $\Omega_1 = \{3,6,7\}$, and $\Omega_2 = \{1,2,4,5,8,9\}$, then the permutation operation is written as follows:

$$\mathbf{R} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The permutation process associated with this partitioning applies as indicated in Figure 1.8.

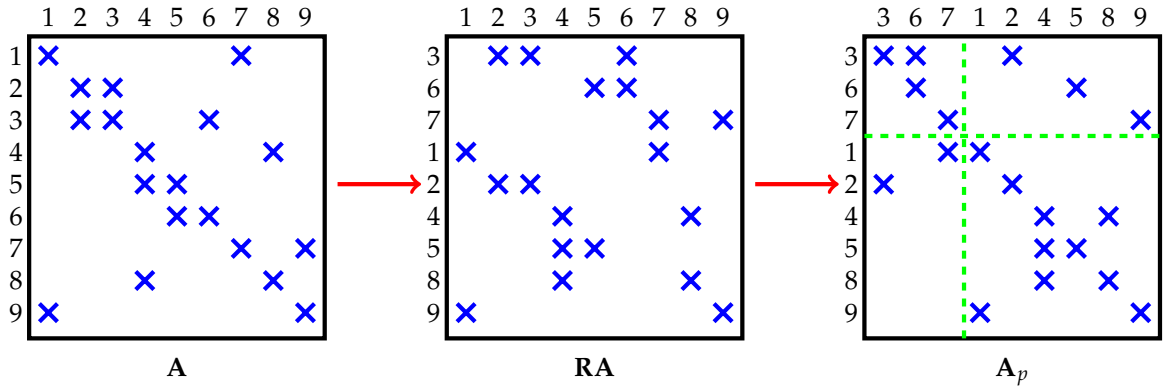


Figure 1.8 – *Row and column permutations after a partitioning.*

After having partitioned the graph and adequately permuted the linear system, we construct the preconditioner before moving subsequently to the last stage, that is, the solution stage. The considered preconditioning options will be dealt with later in this document (Section 1.2.3), in which we will also present the numerical results.

1.2.2.2 Partitioning adjustment based on a posteriori error estimates

Within the context of solving sparse linear systems with an adaptive local choice of preconditioners according to the values of a posteriori error estimates on mesh elements, we model the problem as a double-criteria partitioning problem. It is no longer only a matter of load balacing (criterion n°1), but also of intelligent distribution of data depending on the error level (criterion n°2). For the reasons we have mentioned in Section 1.2.1, and according to an approach that we will explain in detail later in Section 1.2.3, we would like to use the robust preconditioner $\mathbf{M}_S$ for the blocks containing nodes with high values of error estimates, and the less robust preconditioner $\mathbf{M}_W$ for the rest. On the other hand, here we aim to limit as far as possible the use of the preconditioner $\mathbf{M}_S$ as its use is more costly. For this reason, we have decided to distribute

the nodes not only according to the coefficients of the matrix (for computational load balancing or a reduction of dependencies between blocks), but also according to the values of a posteriori errors estimates. This kind of problem can be seen as a dual-criteria problem of graph partitioning, for which we suggest the solution described in Algorithm 12.

This solution consists firstly in constructing the graph of the matrix, then in modifying it on the basis of the values of error estimates so that the vertices corresponding to significant error values will be strongly linked together. As discussed in the beginning of this Section 1.2.2, this means creating heavily weighted edges between those vertices. Otherwise, having modified only the weights of the vertices in proportion to the values of error estimates, there would have been a scattering of the HE (high level of error) nodes on several parts so as to satisfy the load distribution criterion, whereas by modifying the edges' weights, we create artificial connections between the HE nodes by giving them big weights so that they will be kept together in the least number of partitions. For this, we initially locate the highest values of error estimates. To do this, we can either directly specify a minimum threshold (that can be derived from the error estimates theory for example), or fix a minimum percentage (and thus we will have a threshold defined by the product of the percentage and the maximum or the average of error estimates). Then, between these vertices, we are going to create artificial edges that are weighted by the values of error estimates. If there are already existing edges between these vertices, we modify the edge weights so as to favour criterion n°2 over criterion n°1. For this, we start by normalizing all the values of error estimates (by their maximum), then between a vertex i and a vertex j where the respective values of error estimates EE_i and EE_j are significant, we fix the edge weight $w_{i,j}$ at,

$$w_{i,j}^{\textbf{(new)}} = \max_{1 \leq k,l \leq n} w_{k,l}^{\textbf{(old)}} * \boldsymbol{\alpha}_{i,j}; \qquad \boldsymbol{\alpha}_{i,j} := (1 + \frac{EE_i + EE_j}{2}). \tag{1.6}$$

In Formula (1.6) above, we multiply the maximum edge weight of the initial graph by a coefficient that depends on the error values. From the positivity of the values EE_i and EE_j, and their normalization, we can deduce the following inequality for the coefficient $\boldsymbol{\alpha}_{i,j}$,

$$1 < \boldsymbol{\alpha}_{i,j} \leq 2. \tag{1.7}$$

This way, we force the partitioner to take more into account criterion n°2 than criterion n°1. Once the new graph is completed, we go through the same stages of the process detailed in Section 1.2.2.1, partitioning into k parts via a call to "metismex" function of METIS, then building the partition matrix $\mathbf{R}$, and finally the permutation of the linear system. Figure 1.9 represents

Algorithm 12 Partitioning strategy adjusted by a posteriori error estimates

Inputs: $\mathbf{A}$, $\mathbf{EE}$, k
1: Construct the graph that corresponds to the matrix $\mathbf{A}$
2: Normalize $\mathbf{EE}$
3: Locate the vertices S_{HE} corresponding to high values of $\mathbf{EE}$
4: Create artifical EE-weighted edges between the vertices S_{HE} that are not interconnected
5: Replace the edge weights between the already interconnected vertices S_{HE} by greater coefficients involving $\mathbf{EE}$ values
6: Partition into k parts in accordance with the chosen heuristics
7: Build the partition matrix $\mathbf{R}$
8: Permute the linear system by applying $\mathbf{R}$

the partitioning obtained for the matrix considered in the example of Section 1.2.1, after a graph

adjustment with **EE** the vector of error estimates. In this example, we assume that the errors are concentrated on the nodes 6, 10, and 11 of Figure **1.1** for which we consider the following error estimates:

$$EE_{11} = 5 * 10^{-7}, \tag{1.8}$$

$$EE_{10} = 6 * 10^{-7}, \tag{1.9}$$

$$EE_{6} = 7 * 10^{-7}. \tag{1.10}$$

In the graph from Figure **1.9**, the edges in a black solid line correspond to the edges already existing in the initial graph. Here, we assume that the maximum weight of these edges is equal to 1. The edges that were absent in the initial graph and artificially created after the graph adjustment strategy are indicated with red dotted lines whereas the edges in black and red stand for the edges already present in the initial gragh, but whose weights had been modified via Formula (1.6).

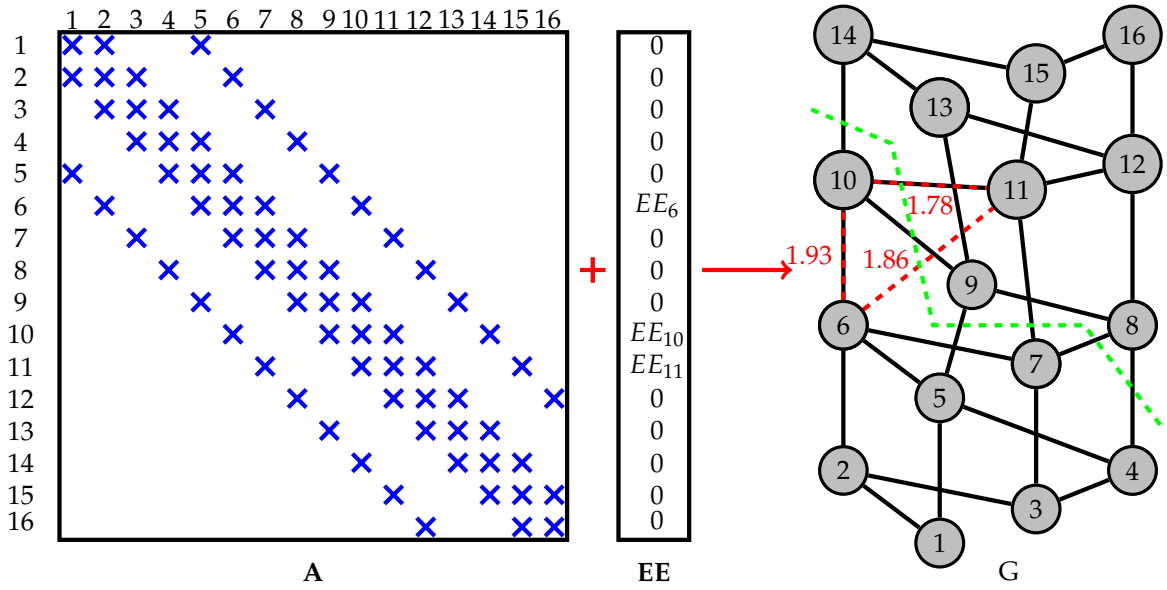


Figure 1.9 – *Constructing and partitioning the graph of the matrix into two parts by taking account of the error estimate values.*

When comparing Figures 1.7 and 1.9, we can notice that because of the high values of error estimates at the nodes 6 and 10, the edge that relates these two nodes is heavily weighted, which made the node 10 shift from the second part to the first, whereas in order to have the same number of nodes in each of the parts, the node 8 was shifted in the opposite direction.

One advantage of this modified strategy of clustering based on error estimates is that it makes it possible not only to have domain partitions that are relatively equal in size (as in the original version), but also to group in the least domains possible the nodes having the most significant error estimates. For a rather limited number of subdomains, it is easier to place all the nodes in a single subdomain with this strategy. While raising considerably the number of domains, we notice that with this modified strategy, we obtain less HE domains (with high errors) than with the classical strategy.

Finally, at this stage the only thing that remains to be done is to construct the adaptive preconditioner in the way described in Section 1.2.3 but this time for the permuted matrix $\mathbf{A}_p$.

1.2.3 Variable block Jacobi-type preconditioning

Error estimates can be good indicators that help identify the parts of the domain where the algebraic error is important and, consequently, it would be logical to provide more means so as to locally improve the quality of computations in these areas. Since the iterative solver cannot allow for such an option, we turned to the preconditioner. Our proposal here is to have a preconditioner with a variable complexity that adapts to the level of error in the different parts of the domain. Thus, the domain decomposition is the most natural option for this kind of problem which is not very far from the context for which the domain decomposition concept was originally initiated, that is, to separately solve problems relating to subdomains and iterate to correct errors.

1.2.3.1 Defining the concept

While in the domain decomposition preconditioning methods mentioned above the subproblems are exactly solved, the Block-Jacobi preconditioner we are considering here in our adaptive approach of local preconditioning is built from inaccurate factorizations, using incomplete block-ILU decompositions on the block diagonal part. Thus, if we consider for example only two LU-type preconditioners ($\mathbf{M}_\mathrm{S}$ for a robust preconditioner, and $\mathbf{M}_\mathrm{W}$ for a less robust preconditioner) like in the global approach of Section 1.1, then we will get a preconditioner in the form $\mathbf{M} = \mathbf{LU}$ with $\mathbf{L}$ and $\mathbf{U}$ that will be constructed block by block, as shown in Figures 2, 3, with the block diagonal factors defined by $[\mathbf{L}_{ii}, \mathbf{U}_{ii}] = \begin{cases} \mathbf{M}_\mathrm{S}(\mathbf{A}_{ii}) & \text{if } \overline{\mathbf{EE}_i} > 0 \\ \mathbf{M}_\mathrm{W}(\mathbf{A}_{ii}) & \text{otherwise} \end{cases}$ where $\overline{\mathbf{EE}_i}$ is the average value of $\mathbf{EE}_i$ which is the subvector of error estimates on subdomain i. For the choice of preconditioner, we have retained ILU(0) decomposition [123], which is the least expensive in computational time, for $\mathbf{M}_\mathrm{W}$, and LU decomposition [5], which is more expensive but yields an exact factorization, for $\mathbf{M}_\mathrm{S}$. We can consider this latter as a direct method for system solving used here to precondition submatrices of subdomains where the average of error estimates is non-zero. The advantage of this block decomposition is that the more subdomains there are, the smaller the size of the submatrices will be. In this way, the cost of LU decomposition remains reasonable.

1.2.3.2 Subdomain merging technique

While applying the adjustment strategy based on a posteriori error estimates, we obtain a permuted matrix $\mathbf{A}_p$ whose block diagonal is formed by submatrices that are ordered according to the error level. However, it should be noted that when we increase the number of subdomains, the eigenvalue bounds (*cf.* Lemma 0.1) are negatively affected. This entails a slower convergence and an increase in the number of iterations. The idea we want to introduce here now aims at exploiting the arrangement of the matrix blocks (by level of error) to merge the subdomains that have nearly the same level of error.
First, this allows us to have less subdomains and thereby less matrix blocks to handle. Secondly, by proceeding this way, it would be possible to compute one preconditioner common to a set of subdomains having the same level of error rather than to compute as many preconditioners

as subdomains, particularly for HE subdomains with high values of error estimates. While the Block-Jacobi strategy of block adaptive preconditioning consists in computing and applying the robust preconditioner several times (for every HE subdomain) on matrices with relatively reduced sizes, the merging technique combined with an adaptive Block-Jacobi preconditioning allows to compute the robust preconditioner once only, but on bigger matrices (because they resulted from the merger of at least two subdomains). Thirdly, such preconditioning would be more accurate since by merging subdomains we compute a preconditioner not only for the diagonal submatrices, but also for a larger block including as well some extradiagonal matrices that represent the connections between subdomains.

In the following, we favour the grouping of the HE subdomains (with high error level) because they are generally very few due to the adjustment of the partitioning based on error estimates. In some cases, this allows to have one single HE block of a reasonable size, and thus limits the use of the robust preconditioner $\mathbf{M}_S$ to only once during the assembly of the adaptive preconditioner.

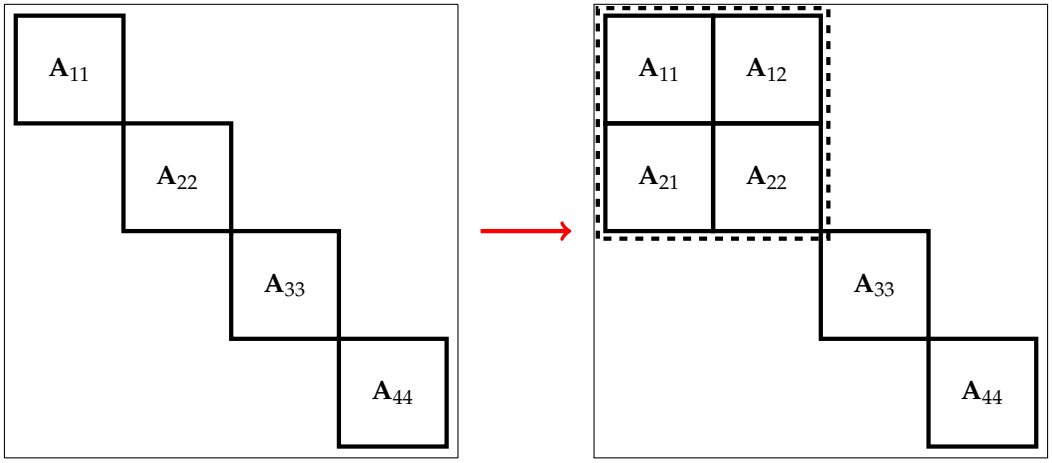


Figure 1.10 – *Shape of the Block-Jacobi preconditioning matrix after the merger of the first two subdomains*

If we reuse the example of Figure 1.7 with 4 subdomains, and assume that the error levels of the first two subdomains are high and close, then the grouping of these two domains is done as illustrated in Figure 1.10. There are 3 local preconditionners to be constructed instead of 4, the first of which is computed for a bigger submatrix, where the off-diagonal matrices $\mathbf{A}_{12}$ and $\mathbf{A}_{21}$ are also taken into account.

1.2.4 Numerical tests

1.2.4.1 Runtime platform

In this part which focuses on the study of the adaptive approach of local preconditioning, we run numerical tests on an Intel Core i7 mainstream laptop equipped with a quad-core processor, with a clock rate of 2.7 Ghz and 16 GB of memory.

1.2.4.2 Computing framework

The adaptive approach of local preconditioning was implemented as prototype in Matlab. The computing parallelization is managed only by Matlab. The linear solver GMRES that we use is already defined as a function in Matlab while the preconditioners are derived from "The matlab suite" collection* of Yousef Saad. The dimension of Krylov subspace for GMRES is set to 100. This work aims to validate the adaptive preconditioning strategies.

1.2.4.3 Presentation of the study cases

As for the numerical results shown later in this section, they were obtained from real data (of real application cases). The linear systems (matrices, second member vectors and error estimates) are extracted from several reservoir simulations. The test cases are the following. SPE10Layer85 is a SPE10 model (see Section 1.1.5) reduced to one horizontal layer (Layer85) characterized by a complex permeability field. This non-stationary model simulates transport on a regular Cartesian grid composed of 60 vertices in X axis and 220 vertices in Y axis. 3DBlackOil is the same model described in Section 1.1.5. For this latter model, we extract the data of two time steps in the middle and in the end of the simulation. We also point out that the matrices of this test case are non-symmetric.

For the computational tests presented here, we consider the same 3DBlackOil test case as in Section 1.2.1 and add a symmetrized version of SPE10Layer85 test case, obtained by constructing the system for the pressure variable only. For this latter test case, the matrix is SPD of size 40040×40040, therefore, the results of Lemmas 0.1 and 0.2 hold.

1.2.4.4 Block preconditioning without permutation

As a first step, we preserve the initial order of the variables, the matrix and other vectors are not permuted. The initial domain is partitioned as explained in Section 1.2.1, and the number of subdomains is varied and we compare the performance (number of solver iterations and the relative residual) obtained with the following preconditioning options:

- **ILU0**: The preconditioner $\mathbf{M}_W = \text{ILU}(0)$ is applied on the entire global matrix.
- **LU**: The LU factorization is applied on the entire global matrix.
- **HE(LU)_LE(LU)**: A Block-Jacobi preconditioner is constructed by computing a preconditioner $\mathbf{M}_S = \text{LU}$ locally on each subdomain.
- **HE(LU)_LE(ILU0)**: A Block-Jacobi preconditioner is constructed according to the adaptive approach by assembling the local preconditioners block by block: by locally computing a preconditioner $\mathbf{M}_S = \text{LU}$ on every (HE) subdomain where the mean value of a posteriori error estimates is significant, and a precondition $\mathbf{M}_S = \text{ILU}(0)$ on every (LE) subdomain where the mean value of a posteriori error estimates is negligible.

It should be mentioned that in the result tables, #Sub indicates the total number of subdomains whereas #HE denotes the number of HE subdomains. For the record, the criterion for sorting

*. http://www-users.cs.umn.edu/ saad/software/home.html

the subdomains in this section reads as follows. A subdomain i is classified as HE subdomain if the average value of a posteriori error estimates on the nodes of this subdomain is nonzero. Otherwise, subdomain i is classified as LE subdomain.

We compare the performances obtained with different preconditioning strategies and with different partition sizes. In addition to the solve times, we consider it is relevant to carry out a computational complexity analysis for the preconditioner construction. We provide estimates of the number of floating-point operations in Tables 1.5, 1.6, 1.7, 1.8, 1.9, 1.10, and 1.11.
Assuming that for a sparse matrix $\mathbf{A}$ of size n derived from the discretization of PDEs on a 3D regular mesh, the computational complexity of an LU factorization is of the same order as a Cholesky factorization, that is $\mathcal{O}(n^2)$ complexity *cf.* [66], and the computational complexity of an ILU(0) factorization is at most equal to $\mathcal{O}(nnz(\mathbf{A}))$, with nnz denoting the number of non-zero coefficients, *cf.* [107, Section 3.3.1], the approximate numbers of floating-point operations needed for computing each preconditioner considered are given by the formulas below:

$$comp(\mathbf{LU}) = \mathcal{O}(n^2) \tag{1.11}$$

$$comp(\mathbf{ILU0}) = \mathcal{O}(nnz(\mathbf{A})) \tag{1.12}$$

$$comp(\mathbf{HE(LU)\_LE(LU)}) = \mathcal{O}(\sum_{1\leq i\leq \#Sub} n_i^2) \tag{1.13}$$

$$comp(\mathbf{HE(LU)\_LE(ILU0)}) = \mathcal{O}(\sum_{i\in HE} n_i^2) + \mathcal{O}(\sum_{j\in LE} nnz(\mathbf{A}_j)) \tag{1.14}$$

Table 1.5 – *The total number of iterations (IT) needed for convergence of preconditioned GMRES for solving the linear system stemming from mid-simulation of 3DBlackOil test case.*

#Sub	#HE	Prec Option	IT	Approx. Prec. Constr. Cost (flops)	Time (sec.)	Rel_Res
1	-	None	1143	-	-	$9.89 * 10^{-7}$
		ILU0	843	$3.26 * 10^5$	30.49	$8.11 * 10^{-7}$
		LU	1	$1.87 * 10^8$	89.39	$2.99 * 10^{-13}$
18	10	HE(LU)_LE(LU)	728	$1.04 * 10^7$	28.71	$9.81 * 10^{-7}$
		HE(LU)_LE(ILU0)	759	$5.88 * 10^6$	24.26	$9.98 * 10^{-7}$
36	15	HE(LU)_LE(LU)	818	$5.20 * 10^6$	27.3	$9.81 * 10^{-7}$
		HE(LU)_LE(ILU0)	855	$2.29 * 10^6$	17.71	$9.96 * 10^{-7}$
72	26	HE(LU)_LE(LU)	941	$2.60 * 10^6$	18.39	$9.98 * 10^{-7}$
		HE(LU)_LE(ILU0)	951	$1.05 * 10^6$	13.94	$9.96 * 10^{-7}$
144	43	HE(LU)_LE(LU)	1037	$1.30 * 10^6$	12.4	$9.93 * 10^{-7}$
		HE(LU)_LE(ILU0)	1037	$4.80 * 10^5$	11.22	$9.93 * 10^{-7}$

With the results obtained (Tables 1.5, 1.6, and 1.7), it can be observed that the adaptive approach of block preconditioning **HE(LU)_LE(ILU0)** allows a faster convergence compared with using one single preconditioner for the entire matrix (**ILU0**, **LU**). Indeed, even though the number of iterations with the adaptive preconditioner is not always lower than that with a LU or ILU(0) preconditioner, we can observe that the total solve time is higher. This is due to the fact that computing complete or incomplete factorization of the entire matrix is rather costly. With respect to a Block-Jacobi preconditioning with the same LU preconditioner for all subdomains **HE(LU)_LE(LU)**, the adaptive preconditioner performs better than **HE(LU)_LE(LU)** in Tables 1.5

Table 1.6 – *The total number of iterations (IT) needed for convergence of preconditioned GMRES for solving the linear system stemming from the end of simulation of 3DBlackOil test case.*

#Sub	#HE	Prec Option	IT	Approx. Prec. Constr. Cost (flops)	Time (sec.)	Rel_Res
1	-	None	961	-	-	$9.92 * 10^{-7}$
		ILU0	76	$3.26 * 10^{5}$	28.74	$9.08 * 10^{-7}$
		LU	1	$1.87 * 10^{8}$	84.53	$5.58 * 10^{-14}$
18	13	HE(LU)_LE(LU)	714	$1.04 * 10^{7}$	27.96	$9.92 * 10^{-7}$
		HE(LU)_LE(ILU0)	726	$7.57 * 10^{6}$	23.27	$9.94 * 10^{-7}$
36	21	HE(LU)_LE(LU)	758	$5.20 * 10^{6}$	25.99	$9.95 * 10^{-7}$
		HE(LU)_LE(ILU0)	763	$3.12 * 10^{6}$	19.38	$9.98 * 10^{-7}$
72	37	HE(LU)_LE(LU)	799	$2.60 * 10^{6}$	15.99	$9.96 * 10^{-7}$
		HE(LU)_LE(ILU0)	800	$1.42 * 10^{6}$	13.59	$9.87 * 10^{-7}$
144	64	HE(LU)_LE(LU)	895	$1.30 * 10^{6}$	10.04	$9.96 * 10^{-7}$
		HE(LU)_LE(ILU0)	895	$6.51 * 10^{5}$	10.63	$9.85 * 10^{-7}$

Table 1.7 – *The total number of iterations (IT) needed for convergence of preconditioned GMRES for solving the linear system stemming from the symmetrized version of SPE10-Layer85 test case.*

#Sub	#HE	Prec Option	IT	Approx. Prec. Constr. Cost (flops)	Time (sec.)	Rel_Res
1	-	None	386 396	-	-	$1 * 10^{-6}$
		ILU0	3 314	$2.9 * 10^{5}$	122.34	$9.99 * 10^{-7}$
		LU	1	$1.60 * 10^{9}$	7527.89	$6.79 * 10^{-11}$
64	32	HE(LU)_LE(LU)	3 261	$2.50 * 10^{7}$	60.85	$9.99 * 10^{-7}$
		HE(LU)_LE(ILU0)	3 261	$1.25 * 10^{7}$	61.87	$9.99 * 10^{-7}$
128	64	HE(LU)_LE(LU)	4 413	$1.25 * 10^{7}$	84.90	$9.99 * 10^{-7}$
		HE(LU)_LE(ILU0)	4 413	$6.31 * 10^{6}$	84.33	$9.99 * 10^{-7}$
256	127	HE(LU)_LE(LU)	7 028	$6.28 * 10^{6}$	129.36	$9.99 * 10^{-7}$
		HE(LU)_LE(ILU0)	7 028	$3.17 * 10^{6}$	128.24	$9.99 * 10^{-7}$
512	254	HE(LU)_LE(LU)	6 048	$3.13 * 10^{6}$	105.89	$9.99 * 10^{-7}$
		HE(LU)_LE(ILU0)	6 048	$1.60 * 10^{6}$	107.81	$9.99 * 10^{-7}$

and 1.6. For Table 1.7) which represents a complex case where the convergence is more difficult to reach, we obtain results that are as good as **HE(LU)_LE(LU)**. We observe also that for the three tables, when the number of subdomains is increased, more iterations are needed for the convergence. This is a characteristic feature of block Jacobi-type methods.

1.2.4.5 Block preconditioning with permutation

In a second phase, we use METIS to partition the domain and permute the linear system accordingly, as described in Section 1.2.2. Henceforth, the new criterion to be used to sort the subdomains is the following. A subdomain i is classified as HE subdomain if the maximum value of a posteriori error estimates on the nodes of this subdomain is greater than a certain percentage δ of the absolute maximum value of a posteriori error estimates: $\max_j (\mathbf{EE_i})_j > \delta \max_{j,k} (\mathbf{EE_k})_j$.

Otherwise, subdomain i is classified as LE subdomain. The reason to opt for this maximum criterion instead of the average criterion is that when we permute the vector of error estimates, the nonzero elements that were initially aggregated in some localized areas become scattered at all the subdomains yielding non-zero error average values on subdomains. The maximum criterion seems to be more suited for this configuration.

Furthermore, if we compare the results of block preconditioning without permutation (Tables 1.6 and 1.7 resp.) and those with permutation (Tables 1.8 and 1.9 resp.), we observe that we manage to lower the number of iterations by using METIS partitioning. In this case, the maximum criterion ensures that the number of HE subdomain remains reasonable in comparison with the unpermuted configuration.

Thereafter, in addition to the preconditioning options (**HE(LU)_LE(LU)** and **HE(LU)_LE(ILU0)**) presented in Section 1.2.4.4, we will introduce new ones that are based on the subdomain merging technique:

- **HE(LU,m)_LE(ILU0)**: A Block-Jacobi preconditioner is computed by computing a preconditioner $\mathbf{M}_S = \text{LU}$ locallly on the merged HE subdomains with significant a posteriori error estimates values (HE: High Error, m: merge), and a preconditioner $\mathbf{M}_W = \text{ILU}(0)$ locally on each LE subdomain with negligible a posteriori error estimates values (LE: Low Error).
- **HE(LU,m)_LE(ILU0,m)**: A Block-Jacobi preconditioner is constructed by computing locally a preconditioner $\mathbf{M}_S = \text{LU}$ on the block resulting from the merger of HE subdomains, and a preconditioner $\mathbf{M}_W = \text{ILU}(0)$ on the block resulting from the merger of LE subdomains.

Remark 1.2. Note that in the case when all HE subdomains are merged together, and all LE subdomains are merged together, the bounds of Lemma0.1 are simplified and the resulting preconditioner $\mathbf{M}$ satisfies:

$$\mathcal{K}(\mathbf{M}^{-1}\mathbf{A}) \leq \frac{1+\gamma}{1-\gamma}; \tag{1.15}$$

where γ is the C.B.S constant for the matrix consisting of the HE block and the LE block on its diagonal and the coupling blocks between those two on the off-diagonal part. And when approximate factorizations are used on one or both blocks to build the preconditioner $\widetilde{\mathbf{M}}$, Lemma 0.2 yields:

$$\mathcal{K}(\widetilde{\mathbf{M}}^{-1}\mathbf{A}) \leq \frac{1+\gamma}{1-\gamma} \times \frac{\max\limits_{1\leq i\leq 2} \alpha_i^{(0)}}{\min\limits_{1\leq j\leq 2} \alpha_j^{(1)}}; \tag{1.16}$$

where $\alpha_i^{(0)}$, $\alpha_i^{(1)}$ are defined by (8) with $(\mathbf{A}_{ii})_{i=1,2}$ standing for the HE block and the LE block respectively.

To conclude this section, we present in Tables 1.8 and 1.10 the numerical results: the number of iterations, the total times and the relative residual for solving the linear system stemming from the final time step of the 3DBlackOil simulation, with all the preconditioning options mentioned before with or without merger of subdomains, after METIS partitioning and permutation of the system. Partitioning is carried out either on the initial graph of the matrix or on the graph adjusted by a posteriori error estimates.

Table 1.8 – *The total number of iterations (IT) needed for convergence of preconditioned GMRES for solving the linear system stemming from the end of simulation of 3DBlackOil test case. The k-way partitioning of METIS was applied here followed by proper permutation of the linear system. The percentage used in the error criterion per subdomain is $\delta = 5\%$.*

#Sub	#HE	Prec Option	IT	Approx. Prec. Constr. Cost (flops)	Time (sec.)	Rel_Res
1	-	None	961	-	-	$9.92 * 10^{-7}$
		ILU0	76	$3.26 * 10^{5}$	28.74	$7.43 * 10^{-7}$
		LU	1	$1.87 * 10^{8}$	84.53	$8.54 * 10^{-13}$
64	11	HE(LU,m)_LE(ILU0)	102	$5.84 * 10^{6}$	5.79	$8.88 * 10^{-7}$
		HE(LU,m)_LE(ILU0,m)	81	$5.89 * 10^{6}$	7.16	$7.46 * 10^{-7}$
		HE(LU)_LE(ILU0)	118	$7.31 * 10^{5}$	4.17	$9.19 * 10^{-7}$
128	13	HE(LU,m)_LE(ILU0)	197	$2.12 * 10^{6}$	6.47	$9.68 * 10^{-7}$
		HE(LU,m)_LE(ILU0,m)	99	$2.20 * 10^{6}$	7.33	$9.86 * 10^{-7}$
		HE(LU)_LE(ILU0)	205	$3.63 * 10^{5}$	5.00	$9.61 * 10^{-7}$
256	19	HE(LU,m)_LE(ILU0)	263	$1.24 * 10^{6}$	5.89	$9.95 * 10^{-7}$
		HE(LU,m)_LE(ILU0,m)	107	$1.34 * 10^{6}$	7.19	$9.38 * 10^{-7}$
		HE(LU)_LE(ILU0)	277	$2.55 * 10^{5}$	5.44	$9.90 * 10^{-7}$
512	32	HE(LU,m)_LE(ILU0)	453	$8.86 * 10^{5}$	7.68	$9.81 * 10^{-7}$
		HE(LU,m)_LE(ILU0,m)	118	$1.03 * 10^{6}$	7.46	$9.38 * 10^{-7}$
		HE(LU)_LE(ILU0)	483	$1.81 * 10^{5}$	6.99	$9.90 * 10^{-7}$

Table 1.9 – *The total number of iterations (IT) needed for convergence of preconditioned GMRES for solving the linear system stemming from the symmetrized version of SPE10-Layer85 test case. The k-way partitioning of METIS was applied here followed by proper permutation of the linear system. The percentage used in the error criterion per subdomain is $\delta = 30\%$*

#Sub	#HE	Prec Option	IT	Approx. Prec. Constr. Cost (flops)	Time (sec.)	Rel_Res
1	-	None	386 396	-	-	$1 * 10^{-6}$
		ILU0	3 314	$2.9 * 10^{5}$	122.34	$9.99 * 10^{-7}$
		LU	1	$1.60 * 10^{9}$	7527.89	$6.79 * 10^{-11}$
64	47	HE(LU,m)_LE(ILU0)	949	$8.69 * 10^{8}$	41.05	$9.97 * 10^{-7}$
		HE(LU,m)_LE(ILU0,m)	948	$8.69 * 10^{8}$	40.14	$9.98 * 10^{-7}$
		HE(LU)_LE(ILU0)	1 330	$1.84 * 10^{7}$	48.39	$9.99 * 10^{-7}$
128	68	HE(LU,m)_LE(ILU0)	1 476	$4.54 * 10^{8}$	39.75	$9.99 * 10^{-7}$
		HE(LU,m)_LE(ILU0,m)	1 499	$4.54 * 10^{8}$	43.71	$9.99 * 10^{-7}$
		HE(LU)_LE(ILU0)	1 391	$6.81 * 10^{6}$	34.69	$9.99 * 10^{-7}$
256	95	HE(LU,m)_LE(ILU0)	1 544	$2.21 * 10^{8}$	32.76	$9.99 * 10^{-7}$
		HE(LU,m)_LE(ILU0,m)	1 441	$2.21 * 10^{8}$	35.89	$9.99 * 10^{-7}$
		HE(LU)_LE(ILU0)	1 669	$2.50 * 10^{6}$	34.36	$9.98 * 10^{-7}$
512	130	HE(LU,m)_LE(ILU0)	2 677	$1.03 * 10^{8}$	50.57	$9.99 * 10^{-7}$
		HE(LU,m)_LE(ILU0,m)	2 718	$1.03 * 10^{8}$	59.09	$9.99 * 10^{-7}$
		HE(LU)_LE(ILU0)	2 611	$9.92 * 10^{5}$	47.79	$9.99 * 10^{-7}$

Table 1.10 – *The total number of iterations (IT) needed for convergence of preconditioned GMRES for solving the linear system stemming from the end of simulation of 3DBlackOil test case. The k-way partitioning of METIS was applied here on the graph adjusted by error estimates, followed by proper permutation of the linear system. The percentage used in the error criterion per subdomain is $\delta = 5\%$*

#Sub	#HE	Prec Option	IT	Approx. Prec. Constr. Cost (flops)	Time (sec.)	Rel_Res
1	-	None	961	-	-	$9.92 * 10^{-7}$
		ILU0	76	$3.26 * 10^{5}$	28.74	$7.43 * 10^{-7}$
		LU	1	$1.87 * 10^{8}$	84.53	$8.54 * 10^{-13}$
64	3	HE(LU,m)_LE(ILU0)	131	$6.73 * 10^{5}$	4.71	$9.68 * 10^{-7}$
		HE(LU,m)_LE(ILU0,m)	98	$7.28 * 10^{5}$	7.17	$9.00 * 10^{-7}$
		HE(LU)_LE(ILU0)	189	$3.94 * 10^{5}$	4.88	$9.77 * 10^{-7}$
128	6	HE(LU,m)_LE(ILU0)	167	$6.37 * 10^{5}$	5.21	$9.74 * 10^{-7}$
		HE(LU,m)_LE(ILU0,m)	101	$7.17 * 10^{5}$	7.48	$9.95 * 10^{-7}$
		HE(LU)_LE(ILU0)	286	$2.97 * 10^{5}$	5.72	$9.83 * 10^{-7}$
256	11	HE(LU,m)_LE(ILU0)	281	$5.56 * 10^{5}$	5.65	$9.72 * 10^{-7}$
		HE(LU,m)_LE(ILU0,m)	118	$6.59 * 10^{5}$	7.49	$9.20 * 10^{-7}$
		HE(LU)_LE(ILU0)	366	$2.38 * 10^{5}$	6.28	$9.90 * 10^{-7}$
512	19	HE(LU,m)_LE(ILU0)	470	$4.25 * 10^{5}$	7.11	$9.83 * 10^{-7}$
		HE(LU,m)_LE(ILU0,m)	132	$5.74 * 10^{5}$	7.58	$9.75 * 10^{-7}$
		HE(LU)_LE(ILU0)	415	$1.77 * 10^{5}$	8.02	$9.69 * 10^{-7}$

Table 1.11 – *The total number of iterations (IT) needed for convergence of preconditioned GMRES for solving the linear system stemming from the symmetrized version of SPE10-Layer85 test case. The k-way partitioning of METIS was applied here on the graph adjusted by error estimates, followed by proper permutation of the linear system. The percentage used in the error criterion per subdomain is $\delta = 60\%$*

#Sub	#HE	Prec Option	IT	Approx. Prec. Constr. Cost (flops)	Time (sec.)	Rel_Res
1	-	None	386 396	-	-	$1 * 10^{-6}$
		ILU0	3 314	$2.9 * 10^{5}$	122.34	$9.99 * 10^{-7}$
		LU	1	$1.60 * 10^{9}$	7527.89	$6.79 * 10^{-11}$
64	6	HE(LU,m)_LE(ILU0)	3 692	$1.48 * 10^{7}$	67.67	$9.99 * 10^{-7}$
		HE(LU,m)_LE(ILU0,m)	4 077	$1.48 * 10^{7}$	88.74	$9.99 * 10^{-7}$
		HE(LU)_LE(ILU0)	3 644	$2.69 * 10^{6}$	67.61	$9.99 * 10^{-7}$
128	10	HE(LU,m)_LE(ILU0)	3 221	$1.06 * 10^{7}$	57.96	$9.99 * 10^{-7}$
		HE(LU,m)_LE(ILU0,m)	2 553	$1.06 * 10^{7}$	59.22	$9.99 * 10^{-7}$
		HE(LU)_LE(ILU0)	3 654	$1.29 * 10^{6}$	65.43	$9.99 * 10^{-7}$
256	17	HE(LU,m)_LE(ILU0)	4 111	$7.74 * 10^{6}$	74.23	$9.99 * 10^{-7}$
		HE(LU,m)_LE(ILU0,m)	3 662	$7.76 * 10^{6}$	79.27	$9.99 * 10^{-7}$
		HE(LU)_LE(ILU0)	4 106	$6.96 * 10^{5}$	73.41	$9.98 * 10^{-7}$
512	59	HE(LU,m)_LE(ILU0)	3 096	$2.24 * 10^{7}$	56.08	$9.99 * 10^{-7}$
		HE(LU,m)_LE(ILU0,m)	3 252	$2.24 * 10^{7}$	69.9	$9.99 * 10^{-7}$
		HE(LU)_LE(ILU0)	4 136	$6.10 * 10^{5}$	74.32	$9.99 * 10^{-7}$

Moreover, we notice that the more we increase the number of domains, the more the complexity

decreases, but on the other hand, we make more iterations (due to block Jacobi). Consequently, in the present case where there is no overlap, the number of subdomains should not be increased by much. Furthermore, we observe that the merger of subdomains allows to decrease the number of iterations compared to the non-merged version. On the other side, the cost of constructing the preconditioner becomes more substantial as we deal with larger blocks. Tables 1.9 and 1.10 show that it can be interesting to pay this cost when the case is complex and the convergence is hard to reach. Another point to outline is that the use of the graph modified by the values of error estimates to partition the matrix leads to a decrease not only in the number of HE domains, but also in the computational complexity of adaptive preconditioners in comparison with the original graph. This is clearly seen when comparing Tables 1.8 and 1.9 with Tables 1.10 and 1.11 respectively.
Lastly, it may be concluded that the adaptive preconditioner **HE(LU)_LE(ILU0)** offers an acceptable solution, with almost as many iterations, as much solve time as **HE(LU)_LE(LU)** preconditioner, and at a lower preconditioning cost. In addition, we recall that the use of an exact factorization (such as LU) can pose a problem of memory insufficiency, and therefore is not always possible on very large matrices such as those we deal with in geoscience simulations.

1.2.5 Conclusion

The main conclusion to draw from this chapter is that error estimates can be considered as an intelligent decision aid tool concerning the choice of the preconditioner. In a first phase, and on the basis of error estimators, we have been able to choose the right preconditioner to be used at every time step during a simulation without any preliminary information about the study case. With the global adaptive algorithm, we realize that overall we have made the right choice of preconditioner even if the gain in time is not always very significant. In a second step, we considered the local algebraic error estimates instead of global ones to derive the local adaptive preconditioning algorithm. The proposed preconditioning strategy shows that performing a specific handling of the blocks where the a posteriori error estimates are significant is interesting. However, the approach suggested is not developed enough. Indeed, a naive variant of Block-Jacobi preconditioner is not sufficient to speed up the solve convergence. Therefore, further research is needed to find a more appropriate preconditioning strategy. In the sequel, we show different approaches that we devised to address this problem.

2

Adaptive solution of linear systems based on a posteriori error estimators

This chapter consists of an article submitted for publication, written with Ani Anciaux-Sedrakian, Laura Grigori, Jan Papež and Soleiman Yousef.

Contents

Abstract

In this chapter, we discuss a new adaptive approach for iterative solution of sparse linear systems arising from partial differential equations (PDE) with self-adjoint operators. The idea is to use the a posteriori estimated local distribution of the algebraic error in order to steer and guide the solve process in such way that the algebraic error is reduced more efficiently in the consecutive iterations. We first explain the motivation behind the proposed procedure and show that it can be equivalently formulated as constructing a special combination of preconditioner and initial guess for the original system. We present some numerical experiments in order to identify when the adaptive procedure can be of practical use.
Keywords— Algebraic error, adaptivity, iterative solve, preconditioning, domain decomposition

2.1 Introduction

The seminal as well as recent results on a posteriori error estimation allowed various adaptive concepts in numerical solution of partial differential equations (PDEs). For instance, an a posteriori local estimation of the discretization error (see, e.g., [14, 141, 30]) forms the basis for an adaptive mesh refinement. Such refinement can reduce the norm of the discretization error at a significantly lower cost in comparison to uniform mesh refinement, and typically results in a close-to-uniform spatial distribution of the error over the domain, see, e.g. [112]. These types of estimators, however, typically assume the exact solution of the associated algebraic system that is impossible to achieve in practice.

Inclusion of an inexact (approximate) algebraic solution into error estimators gave rise to *inexact* adaptive solution procedures, which, as a crucial ingredient, involve stopping criteria for iterative algebraic solvers, see, e.g., [10, Section 4]. The corresponding error estimators are typically decomposed into several parts that are identified with different components of the overall error, such as linearization, discretization and algebraic. The criteria in literature are based on well justified heuristics (see, e.g., [19, 55]) and, recently in [113, 114], also on mathematically rigorous proofs.

A common drawback of the above mentioned, rigorously justified estimators is their evaluation cost, which is typically (very) high with respect to the cost of an algebraic solver iteration. However, recent work [144] has resulted in the development of a posteriori estimates that can be easily coded, cheaply evaluated, and efficiently used in practical simulations providing a guaranteed control over different error components. It has confirmed that the computation of error estimators can be accessible even within non-academic contexts.

In this chapter, we introduce a novel adaptive preconditioner for iteratively solving sparse linear systems arising from PDEs that modifies the iteration process according to the a posteriori estimated local distribution of the *algebraic error*. To the best of our knowledge, there are yet no such procedures described in the literature. This chapter therefore opens a discussion if adaptive approaches aiming at reducing the algebraic error in targeted parts of the domain are worth considering (at least in some cases) and how this aim can be achieved. In this chapter we focus on self-adjoint PDE problems of second order only.

The adaptive procedure proposed in this chapter can be briefly described as follows. In a given iteration step, based on the algebraic error distribution, a part of the solution domain and the

associated algebraic degrees of freedom with high algebraic error are indicated. Then a block matrix splitting is introduced and used in a partitioned matrix procedure in order to yield, in the consecutive iterations, the residual vectors vanishing in the degrees of freedom indicated in the first step. We show that the proposed procedure corresponds to building, in a posteriori fashion based on information on the algebraic error at the above-mentioned iteration step, a particular combination of preconditioner and initial guess for following iterations. In addition, the sufficient and necessary conditions for attaining vanishing residuals are discussed.

This chapter is organized as follows. Section 2.2 presents the model problem. Then, for that model, Section 2.3 briefly recalls one way to estimate the local distribution of the algebraic error using flux reconstructions. In Section 2.4, we introduce a matrix splitting based on the distribution of algebraic errors. In Section 2.5, we propose the above mentioned adaptive procedure. In Section 2.6, we present and comment several numerical experiments before reaching a conclusion on when this procedure can be useful in accelerating the iterative solver. Finally, the conclusion overviews the work undertaken in this research and outlines directions for future study.

2.2 Model problem

This section introduces the model problem, and presents the key assumption that motivates the need for an adaptive solving procedure. This section and the following one (2.2 and 2.3) then hold for the next chapters(3 and 4).

Let $\Omega \subset \mathbb{R}^d$, $1 \leq d \leq 3$ be a polytopal domain (open, bounded and connected set). We denote by $\overline{\Omega}$, Ω^o, $\partial\Omega$ and $\mathcal{T}_h$ resp. the closure, interior, boundary and a matching simplicial mesh of Ω. The extension of the results to nonmatching meshes is possible. We use the standard notation $L^2(\Omega)$, $H^1(\Omega)$ and $H^1_0(\Omega)$ for the spaces of integrable functions, resp. integrable functions admitting weak derivations, and trace vanishing on $\partial\Omega$. For a vector w of length $n \in \mathbb{N}$ and a subset $L \subset [\![1, n]\!]$, we denote by w_L the restriction of w to its components whose indexes belong to L.

Consider the problem that consists in seeking $u : \Omega \to \mathbb{R}$ such that:

$$\begin{cases} -\nabla \cdot (\underline{\mathbf{K}} \nabla \underline{u}) = \underline{f} & \text{in } \Omega \\ \underline{u} = 0 & \text{on } \partial\Omega \end{cases} \tag{2.1}$$

where $\underline{f} : \Omega \to \mathbb{R}$ is a source term in $L^2(\Omega)$, and $\underline{\mathbf{K}}$ is an uniformly bounded and positive definite diffusion tensor. For the sake of simplicity we assume that $\underline{f}$ and $\underline{\mathbf{K}}$ are piecewise constant with respect to the mesh $\mathcal{T}_h$. The weak form reads, find $\underline{u} \in V := H^1_0(\Omega)$ such that

$$a(\underline{u}, \underline{v}) := (\underline{\mathbf{K}}^{\frac{1}{2}} \nabla \underline{u}, \underline{\mathbf{K}}^{\frac{1}{2}} \nabla \underline{v}) = (\underline{f}, \underline{v}) \qquad \forall \underline{v} \in V \tag{2.2}$$

where a is a bilinear form. Associated to $\mathcal{T}_h$, let there be a discrete subspace $V_h \subset V$. The Galerkin solution $u_h \in V_h$ satisfies

$$(\underline{\mathbf{K}}^{\frac{1}{2}} \nabla \underline{u}_h, \underline{\mathbf{K}}^{\frac{1}{2}} \nabla \underline{v}_h) = (\underline{f}, \underline{v}_h) \qquad \forall v_h \in V_h. \tag{2.3}$$

Considering a basis $(\varphi_l)_{1 \leq l \leq n}$ of V_h, this problem is equivalent to a system of linear algebraic

equations,

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}, \tag{2.4}$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a symmetric positive definite (SPD) matrix defined by $\mathbf{A}_{jk} = (\underline{\mathbf{K}}^{\frac{1}{2}} \nabla \varphi_k, \underline{\mathbf{K}}^{\frac{1}{2}} \nabla \varphi_j)$, $1 \leq j,k \leq n$, and $\mathbf{b} \in \mathbb{R}^n$ is the right hand side vector, $\mathbf{b}_j = (\underline{f}, \varphi_j)$. The continuous solution is then given by $\underline{u}_h = \sum_{l=1}^{n} x_l \varphi_l$. Let $\mathbf{x}^{(i)}$ be an approximate solution of (2.4) obtained after running i iterations of an iterative solver, and $\underline{u}_h^{(i)} = \sum_{l=1}^{n} x_l^{(i)} \varphi_l$ the associated function from V_h. We denote by $\mathbf{r}^{(i)}$ the corresponding algebraic residual vector $\mathbf{r}^{(i)} := \mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(i)}$. A relevant measure of the algebraic error is the energy norm

$$\|\underline{\mathbf{K}}^{\frac{1}{2}} \nabla (\underline{u}_h - \underline{u}_h^{(i)})\|_{\mathrm{L}^2(\Omega)} = \sqrt{a(\underline{u}_h - \underline{u}_h^{(i)}, \underline{u}_h - \underline{u}_h^{(i)})} = \|\mathbf{x} - \mathbf{x}^{(i)}\|_{\mathbf{A}} = \|\mathbf{r}^{(i)}\|_{\mathbf{A}^{-1}}. \tag{2.5}$$

The adaptive solution of linear systems proposed in this work is based on the fact that we can (tightly) estimate the local distribution of the error

$$\|\underline{\mathbf{K}}^{\frac{1}{2}} \nabla (\underline{u}_h - \underline{u}_h^{(i)})\|_{\mathrm{L}^2(K)}, \qquad \forall K \in \mathcal{T}_h, \tag{2.6}$$

where K typically stands for the mesh elements ($\overline{\Omega} = \cup \overline{K}$). Based on this, we decompose the domain Ω into two disjoint parts Ω_1 and Ω_2:

$$\begin{cases} \overline{\Omega}_1 \cup \overline{\Omega}_2 & = \overline{\Omega} \\ \Omega_1^{\mathrm{o}} \cap \Omega_2^{\mathrm{o}} & = \emptyset \end{cases} \tag{2.7}$$

where Ω_1 is the part with the high algebraic error:

$$\boxed{||\underline{\mathbf{K}}^{1/2} \nabla (\underline{u}_h - \underline{u}_h^{(i)})||^2_{\mathrm{L}^2(\Omega_1)} \gg ||\underline{\mathbf{K}}^{1/2} \nabla (\underline{u}_h - \underline{u}_h^{(i)})||^2_{\mathrm{L}^2(\Omega_2)}} \tag{2.8}$$

In fact, (2.8) is our main starting hypothesis. Figure 2.1 gives an illustrative example with Ω_1 and Ω_2 composed of a single element each.

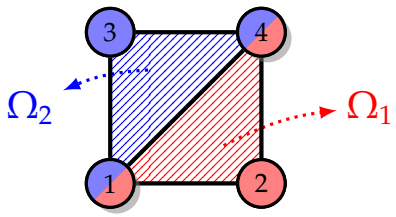


Figure 2.1 – *Simple example of the decomposition* (2.7) *with a* 2×2 *mesh grid.*

2.3 A posteriori error estimates

In this section we describe briefly one approach to estimate the local distribution of the algebraic error (2.6) using flux reconstructions following [79, 55, 114] and references therein. We start by introducing the basic techniques of these a posteriori error estimates, then we detail how

to get a sharp computable upper bound on the algebraic error. Note that this section recalls existing techniques and results and we only adapt the a posteriori error estimate of [114] to our chosen model problem.

2.3.1 Basic a posteriori error estimates

Assuming that a Galerkin solution u_h is available, we start by bounding the energy norm of the error $\underline{u} - \underline{u}_h$ represented as

$$\|\underline{\mathbf{K}}^{\frac{1}{2}}\nabla(\underline{u} - \underline{u}_h)\|_{\mathrm{L}^2(\Omega)} = \sup_{\underline{v}\in V, \|\nabla\underline{v}\|=1} (\underline{\mathbf{K}}^{\frac{1}{2}}\nabla(\underline{u} - \underline{u}_h), \underline{\mathbf{K}}^{\frac{1}{2}}\nabla\underline{v}). \tag{2.9}$$

Note that following (2.2), then

$$\|\underline{\mathbf{K}}^{\frac{1}{2}}\nabla(\underline{u} - \underline{u}_h)\|_{\mathrm{L}^2(\Omega)} = \sup_{\underline{v}\in V, \|\nabla\underline{v}\|=1} \{(\underline{f}, \underline{v}) - (\underline{\mathbf{K}}^{\frac{1}{2}}\nabla\underline{u}_h, \underline{\mathbf{K}}^{\frac{1}{2}}\nabla\underline{v})\}. \tag{2.10}$$

The key ingredient of our estimate is a reconstructed flux $\boldsymbol{\theta}_h$ which is a piecewise polynomial function in the Raviart–Thomas–Nédélec subspace $\mathbf{RTN}(\mathcal{T}_h)$ of the infinite-dimensional space $\mathbf{H}(\mathrm{div}, \Omega)$ which is reconstructed in order to mimic the continuous flux $\boldsymbol{\theta} := -\underline{\mathbf{K}}\nabla\underline{u}$. In other words, $\boldsymbol{\theta}_h$ is reconstructed to satisfy

$$\nabla \cdot \boldsymbol{\theta}_h = \underline{f}. \tag{2.11}$$

Recall from Section 2.2 that $\underline{f}$ is assumed to be piecewise constant with respect to the mesh $\mathcal{T}_h$. Now, we use the Green and the Cauchy–Schwarz inequality together with (2.10) and we follow [114, Section 4.1], to write

$$\begin{aligned}
\|\underline{\mathbf{K}}^{\frac{1}{2}}\nabla(\underline{u} - \underline{u}_h)\|_{\mathrm{L}^2(\Omega)} &= \inf_{\substack{\boldsymbol{\sigma}\in\mathbf{H}(\mathrm{div},\Omega)\\ \nabla\cdot\boldsymbol{\sigma}=\underline{f}}} \sup_{\substack{\underline{v}\in V\\ \|\nabla\underline{v}\|=1}} \{(\underline{f} - \nabla\cdot\boldsymbol{\sigma}, \underline{v}) \\
&\qquad\qquad -(\underline{\mathbf{K}}^{\frac{1}{2}}\nabla\underline{u}_h + \underline{\mathbf{K}}^{-\frac{1}{2}}\boldsymbol{\sigma}, \underline{\mathbf{K}}^{\frac{1}{2}}\nabla\underline{v})\} \\
&= \inf_{\substack{\boldsymbol{\sigma}\in\mathbf{H}(\mathrm{div},\Omega)\\ \nabla\cdot\boldsymbol{\sigma}=\underline{f}}} \sup_{\substack{\underline{v}\in V\\ \|\nabla\underline{v}\|=1}} \{-(\underline{\mathbf{K}}^{\frac{1}{2}}\nabla\underline{u}_h + \underline{\mathbf{K}}^{-\frac{1}{2}}\boldsymbol{\sigma}, \underline{\mathbf{K}}^{\frac{1}{2}}\nabla\underline{v})\} \\
&= \inf_{\substack{\boldsymbol{\sigma}\in\mathbf{H}(\mathrm{div},\Omega)\\ \nabla\cdot\boldsymbol{\sigma}=\underline{f}}} \|\underline{\mathbf{K}}^{\frac{1}{2}}\nabla\underline{u}_h + \underline{\mathbf{K}}^{-\frac{1}{2}}\boldsymbol{\sigma}\|_{\mathrm{L}^2(\Omega)} \\
&\leq \|\underline{\mathbf{K}}^{\frac{1}{2}}\nabla\underline{u}_h + \underline{\mathbf{K}}^{-\frac{1}{2}}\boldsymbol{\theta}_h\|_{\mathrm{L}^2(\Omega)}.
\end{aligned}$$

This gives a guaranteed upper bound on the discretization error. Note that the obtained estimate relies only on the weak formulation and the reconstructed flux $\boldsymbol{\theta}_h$. Finally, it is important to mention that the needed reconstructed flux $\boldsymbol{\theta}_h$ can be easily reconstructed for various discretization schemes like finite elements, nonconforming finite elements, discontinuous Galerkin, finite volumes, and mixed finite elements, see [55] for more details.

2.3.2 Upper bound on the algebraic error

In this section we suppose that we use an iterative solver to obtain an approximate solution $\underline{u}_h^{(i)}$ of (2.3) after running i iterations. In order to estimate the algebraic error we first introduce a representation of the algebraic residual vector which will be a function $\underline{s}_h^{(i)} \in \mathrm{L}^2(\Omega)$ satisfying

$$(\underline{s}_h^{(i)}, \varphi_j) = \mathbf{r}_j^{(i)}, \qquad 1 \leq j \leq n. \tag{2.12}$$

Details about the reconstruction of $\underline{s}_h^{(i)}$ with two differents examples can be found in [114, Section 5.1]. Note that from (2.12) and the definition of the algebraic residual vector $\mathbf{r}^{(i)}$ one can write

$$(\underline{s}_h^{(i)}, \varphi_j) = (\underline{f}, \varphi_j) - (\underline{\mathbf{K}}^{\frac{1}{2}} \nabla \underline{u}_h^{(i)}, \underline{\mathbf{K}}^{\frac{1}{2}} \nabla \varphi_j), \qquad 1 \leq j \leq n. \tag{2.13}$$

Consequently,

$$(\underline{s}_h^{(i)}, \underline{v}_h) = (\underline{f}, \underline{v}_h) - (\underline{\mathbf{K}}^{\frac{1}{2}} \nabla \underline{u}_h^{(i)}, \underline{\mathbf{K}}^{\frac{1}{2}} \nabla \underline{v}_h). \tag{2.14}$$

Using (2.3), then (2.14) gives

$$(\underline{s}_h^{(i)}, \underline{v}_h) = (\underline{\mathbf{K}}^{\frac{1}{2}} (\nabla \underline{u}_h - \nabla \underline{u}_h^{(i)}), \underline{\mathbf{K}}^{\frac{1}{2}} \nabla \underline{v}_h). \tag{2.15}$$

This representation of the algebraic residual vector plays a key role in the estimation of the algebraic error. By applying the Cauchy-Schwarz inequality together with the Friedrichs inequality on (2.15), one gets

$$\begin{aligned}
(\underline{\mathbf{K}}^{\frac{1}{2}} (\nabla \underline{u}_h - \nabla \underline{u}_h^{(i)}), \underline{\mathbf{K}}^{\frac{1}{2}} \nabla \underline{v}_h) &= (\underline{s}_h^{(i)}, \underline{v}_h) \\
&\leq \|\underline{s}_h^{(i)}\|_{\mathrm{L}^2(\Omega)} \|\underline{v}_h\|_{\mathrm{L}^2(\Omega)} \\
&\leq \|\underline{s}_h^{(i)}\|_{\mathrm{L}^2(\Omega)} \left(C_{\mathrm{F}} h_\Omega \|\nabla \underline{v}_h\|_{\mathrm{L}^2(\Omega)} \right) \\
&\leq \|\underline{s}_h^{(i)}\|_{\mathrm{L}^2(\Omega)} \left(C_{\mathrm{F}} h_\Omega \lambda_{\underline{\mathbf{K}}}^{-\frac{1}{2}} \|\underline{\mathbf{K}}^{\frac{1}{2}} \nabla \underline{v}_h\|_{\mathrm{L}^2(\Omega)} \right),
\end{aligned}$$

where $0 < C_{\mathrm{F}} \leq 1$ is the constant from the Friedrichs inequality, h_Ω the diameter of the domain Ω, and $\lambda_{\underline{\mathbf{K}}}$ the smallest eigenvalue of $\underline{\mathbf{K}}$. First computable upper bound is then obtained for the algebraic error as

$$\|\underline{\mathbf{K}}^{\frac{1}{2}} \nabla (\underline{u}_h - \underline{u}_h^{(i)})\|_{\mathrm{L}^2(\Omega)} \leq C_{\mathrm{F}} h_\Omega \lambda_{\underline{\mathbf{K}}}^{-\frac{1}{2}} \|\underline{s}_h^{(i)}\|_{\mathrm{L}^2(\Omega)}. \tag{2.16}$$

However, this upper bound often yields a significant overestimation; see, e.g., [114, Sections 3.1 and 4.2]. An improvement of the upper bound (2.16) can be obtained by using flux reconstruction techniques and additional algebraic iterations. Following Section 2.3.1 and [114, Section 5.3], we consider a reconstructed flux $\boldsymbol{\theta}_h^{(i)} \in \mathbf{RTN}(\mathcal{T}_h)$ satisfying $\nabla \cdot \boldsymbol{\theta}_h^{(i)} = \underline{f} - \underline{s}_h^{(i)}$. Then, after $\nu > 0$ additional iterations we similarly construct from the algebraic residual vector $r^{(i+\nu)}$ a representation $\underline{s}_h^{(i+\nu)}$, and another reconstructed flux $\boldsymbol{\theta}_h^{(i+\nu)} \in \mathbf{RTN}(\mathcal{T}_h)$ satisfying $\nabla \cdot \boldsymbol{\theta}_h^{(i+\nu)} = \underline{f} - \underline{s}_h^{(i+\nu)}$. With these different reconstructions we have

$$\underline{s}_h^{(i)} = \underline{s}_h^{(i+\nu)} + \nabla \cdot \boldsymbol{\theta}_h^{(i+\nu)} - \nabla \cdot \boldsymbol{\theta}_h^{(i)},$$

and therefore (2.15) gives

$$(\underline{\mathbf{K}}^{\frac{1}{2}}(\nabla \underline{u}_h - \nabla \underline{u}_h^{(i)}), \underline{\mathbf{K}}^{\frac{1}{2}} \nabla \underline{v}_h) = (\underline{\mathbf{K}}^{-\frac{1}{2}}(\boldsymbol{\theta}_h^{(i+\nu)} - \boldsymbol{\theta}_h^{(i)}), \underline{\mathbf{K}}^{\frac{1}{2}} \nabla \underline{v}_h) + (\underline{s}_h^{(i+\nu)}, \underline{v}_h).$$

Consequently,

$$\|\underline{\mathbf{K}}^{\frac{1}{2}} \nabla (\underline{u}_h - \underline{u}_h^{(i)})\|_{\mathrm{L}^2(\Omega)} \leq \|\underline{\mathbf{K}}^{-\frac{1}{2}}(\boldsymbol{\theta}_h^{(i+\nu)} - \boldsymbol{\theta}_h^{(i)})\|_{\mathrm{L}^2(\Omega)} + C_{\mathrm{F}} h_\Omega \lambda_{\underline{\mathbf{K}}}^{-\frac{1}{2}} \|\underline{s}_h^{(i+\nu)}\|_{\mathrm{L}^2(\Omega)} \tag{2.17}$$

The idea of using additional algebraic iterations is very useful in practice [61]. In fact, for a sufficiently large value of ν one can assume that there exists $\gamma > 0$ such that

$$C_{\mathrm{F}} h_\Omega \lambda_{\underline{\mathbf{K}}}^{-\frac{1}{2}} \|\underline{s}_h^{(i+\nu)}\|_{\mathrm{L}^2(\Omega)} \leq \gamma \|\underline{\mathbf{K}}^{-\frac{1}{2}}(\boldsymbol{\theta}_h^{(i+\nu)} - \boldsymbol{\theta}_h^{(i)})\|_{\mathrm{L}^2(\Omega)},$$

so that

$$\|\underline{\mathbf{K}}^{\frac{1}{2}} \nabla (\underline{u}_h - \underline{u}_h^{(i)})\|_{\mathrm{L}^2(\Omega)} \leq (1+\gamma) \|\underline{\mathbf{K}}^{-\frac{1}{2}}(\boldsymbol{\theta}_h^{(i+\nu)} - \boldsymbol{\theta}_h^{(i)})\|_{\mathrm{L}^2(\Omega)},$$

which gives an upper bound easily computed and cheaply evaluated in practice even for complex problems, see [144] for details.

Remark 2.1 (Local indicators for the algebraic error). In order to estimate the local distribution of the algebraic error (2.6) using flux reconstructions, one can use the local indicators $\eta_{\mathrm{alg},K}^{(i)} := \|\underline{\mathbf{K}}^{-\frac{1}{2}}(\boldsymbol{\theta}_h^{(i+\nu)} - \boldsymbol{\theta}_h^{(i)})\|_{\mathrm{L}^2(K)} + C_{\mathrm{F}} h_\Omega \lambda_{\underline{\mathbf{K}}}^{-\frac{1}{2}} \|\underline{s}_h^{(i+\nu)}\|_{\mathrm{L}^2(K)}$. Relying on the previous discussion, in practice with a sufficiently large ν one can use the local algebraic indicator $\|\underline{\mathbf{K}}^{-\frac{1}{2}}(\boldsymbol{\theta}_h^{(i+\nu)} - \boldsymbol{\theta}_h^{(i)})\|_{\mathrm{L}^2(K)}$ which can be the ingredient of our adaptive procedure.

Remark 2.2 (A posteriori error estimates for the total error). A computable upper bound can be obtained on the energy norm of the total error $\underline{u} - \underline{u}_h^{(i)}$ following the same ideas as in Section 2.3.1 and Section 2.3.2,

$$\|\underline{\mathbf{K}}^{\frac{1}{2}} \nabla (\underline{u} - \underline{u}_h^{(i)})\|_{\mathrm{L}^2(\Omega)} \leq \eta_{\mathrm{dis}}^{(i)} + \eta_{\mathrm{alg}}^{(i)},$$

with

$$\eta_{\mathrm{dis}}^{(i)} := \|\underline{\mathbf{K}}^{\frac{1}{2}} \nabla \underline{u}_h^{(i)} + \underline{\mathbf{K}}^{-\frac{1}{2}} \boldsymbol{\theta}_h^{(i)}\|_{\mathrm{L}^2(\Omega)}$$

and

$$\eta_{\mathrm{alg}}^{(i)} := \|\underline{\mathbf{K}}^{-\frac{1}{2}}(\boldsymbol{\theta}_h^{(i+\nu)} - \boldsymbol{\theta}_h^{(i)})\|_{\mathrm{L}^2(\Omega)} + C_{\mathrm{F}} h_\Omega \lambda_{\underline{\mathbf{K}}}^{-\frac{1}{2}} \|\underline{s}_h^{(i+\nu)}\|_{\mathrm{L}^2(\Omega)},$$

see [114] for the full demonstration.

Remark 2.3 (A posteriori error estimates in the multilevel setting). There is also another way to construct upper bounds for the total and algebraic errors without the need of running additional iterations of the algebraic solver. The construction assumes the existence of the hierarchy of meshes, with a global solve on the coarsest mesh; see [113] for more details.

2.4 Matrix decomposition and local error reduction

In this section, we introduce a sum splitting of the matrix $\mathbf{A}$ associated to the partitioning (2.7). Then, we propose a preconditioner that ensures that the algebraic error is locally decreased on the targeted subdomain.

2.4.1 Matrix decomposition: sum splitting

According to the domain decomposition of (2.7) mentioned above, we denote by $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$ the local stiffness matrices for the subdomains Ω_1 and Ω_2, respectively. While for the matrix $\mathbf{A}$ we have

$$\mathbf{A}_{jk} = (\underline{\mathbf{K}}^{\frac{1}{2}} \nabla \varphi_k, \underline{\mathbf{K}}^{\frac{1}{2}} \nabla \varphi_j), \qquad 1 \leq j,k \leq n \ ,$$

we define

$$\mathbf{A}^{(1)}_{jk} = (\underline{\mathbf{K}}^{\frac{1}{2}} \nabla \varphi_k, \underline{\mathbf{K}}^{\frac{1}{2}} \nabla \varphi_j)_{\Omega_1},\ 1 \leq j,k \leq n,\ \text{supp}\ \varphi_k \cap \Omega_1 \neq \emptyset,\ \text{supp}\ \varphi_j \cap \Omega_1 \neq \emptyset,$$

$$\mathbf{A}^{(2)}_{jk} = (\underline{\mathbf{K}}^{\frac{1}{2}} \nabla \varphi_k, \underline{\mathbf{K}}^{\frac{1}{2}} \nabla \varphi_j)_{\Omega_2},\ 1 \leq j,k \leq n,\ \text{supp}\ \varphi_k \cap \Omega_2 \neq \emptyset,\ \text{supp}\ \varphi_j \cap \Omega_2 \neq \emptyset.$$

For the ease of presentation we assume a convenient ordering such that the variables corresponding to the vertices of Ω_1 are sorted first and those of Ω_2 second. Then we can split the original operator, represented algebraically by the stiffness matrix $\mathbf{A}$, as follows

$$\mathbf{A} = \mathbf{A}^{(1)}_p + \mathbf{A}^{(2)}_p, \tag{2.18}$$

where $\mathbf{A}^{(1)}_p, \mathbf{A}^{(2)}_p$ are symmetric positive semidefinite (SPSD) defined as:

$$\mathbf{A}^{(1)}_p = \left(\begin{array}{c|c} \mathbf{A}^{(1)} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right); \qquad \mathbf{A}^{(2)}_p = \left(\begin{array}{c|c} \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{A}^{(2)} \end{array}\right).$$

They are the extensions of the local stiffness matrices $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$ to the entire domain. Then, we get the equivalent formulation of (2.8) in the matrix representation,

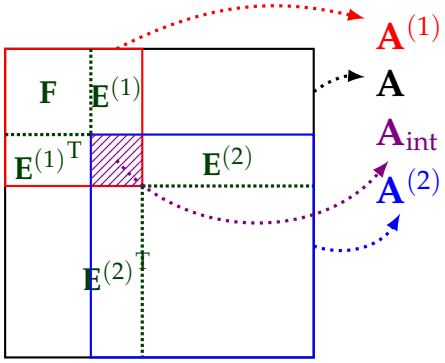
$$\boxed{(\mathbf{x} - \mathbf{x}^{(i)})^{\mathrm{T}} \cdot \mathbf{A}^{(1)}_p \cdot (\mathbf{x} - \mathbf{x}^{(i)}) \gg (\mathbf{x} - \mathbf{x}^{(i)})^{\mathrm{T}} \cdot \mathbf{A}^{(2)}_p \cdot (\mathbf{x} - \mathbf{x}^{(i)})}, \tag{2.19}$$

where $\mathbf{x}^{(i)}$ is the approximate solution at iteration i. Figure 2.2(a) illustrates how the global matrix $\mathbf{A}$ is built from $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$. The shaded part represents the common vertices between $\overline{\Omega}_1$ and $\overline{\Omega}_2$. It is the part of the matrix where the contributions from both subdomains are summed together.

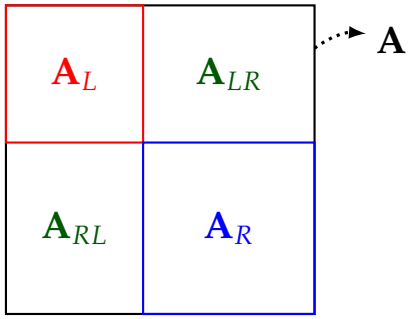
2.4.2 Matrix decomposition: Block partitioning

In this section, we derive a 2×2 block-partitioning of the matrix. This enables us to define a second approach based on appropriate initial guess and preconditioners to reduce the global error and make the residual nil in Ω_1. In general, unless $\partial\Omega_1 \cap \partial\Omega \neq \emptyset$, the matrix $\mathbf{A}^{(1)}$ is singular. Since many common preconditioners and algebraic techniques (such as Cholesky factorization) are not suitable for a singular matrix, we replace the sum-splitting of the operator, as in (2.18), by a block partitioning of the matrix, such as

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_L & \mathbf{A}_{LR} \\ \mathbf{A}_{RL} & \mathbf{A}_R \end{pmatrix}. \tag{2.20}$$



(a) Splitting of matrix $\mathbf{A}$ with local stiffness matrices



(b) Algebraic 2×2 block splitting of matrix $\mathbf{A}$

Figure 2.2 – *Splittings of matrix* $\mathbf{A}$ *with local stiffness matrices (left) and the associated algebraic* 2×2 *block splitting (right)*

Now if we decompose $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$ as:

$$\mathbf{A}^{(1)} = \begin{pmatrix} \mathbf{F} & \mathbf{E}^{(1)} \\ \mathbf{E}^{(1)^{\mathrm{T}}} & \mathbf{A}_{\mathrm{int}}^{(1)} \end{pmatrix}; \qquad \mathbf{A}^{(2)} = \begin{pmatrix} \mathbf{A}_{\mathrm{int}}^{(2)} & \mathbf{E}^{(2)} \\ \mathbf{E}^{(2)^{\mathrm{T}}} & \mathbf{A}_R \end{pmatrix}, \tag{2.21}$$

then the algebraic 2×2 block splitting of Figure 2.2(b) is built as follows:

$$\mathbf{A}_{LR} = \mathbf{A}_{RL}^{\mathrm{T}} = \begin{pmatrix} \mathbf{0} \\ \mathbf{E}^{(2)} \end{pmatrix}; \qquad \mathbf{A}_L = \mathbf{A}^{(1)} + \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\mathrm{int}}^{(2)} \end{pmatrix}, \tag{2.22}$$

where we denote,

- L: the set of nodes that belong to Ω_1,
- R: the complementary of L.

Clearly, the matrix $\mathbf{A}_R$ does not contain any information on the common degrees of freedom, since the shaded part ($\mathbf{A}_{\mathrm{int}} = \mathbf{A}_{\mathrm{int}}^{(1)} + \mathbf{A}_{\mathrm{int}}^{(2)}$) is fully and exclusively integrated in $\mathbf{A}_L$. Note also that the matrices $\mathbf{A}_L$ and $\mathbf{A}_R$ are symmetric positive definite.

Remark 2.4. The number of degrees of freedom of the overlapping part $\mathbf{A}_{\mathrm{int}}$ depends on the algebraic error distribution. It may not be small with respect to the sizes of $\mathbf{A}_L$ and $\mathbf{A}_R$ respectively.

Splitting the vectors $\mathbf{b}$ and $\mathbf{x}$ according to the partitioning in (2.20) yields the vectors $\mathbf{b}_L$, $\mathbf{b}_R$, $\mathbf{x}_L$ and $\mathbf{x}_R$. Then, the corresponding block formulas for the solution $\mathbf{x}$ and the residual for $\mathbf{x}^{(i)}$ are:

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b} \iff \begin{pmatrix} \mathbf{A}_L & \mathbf{A}_{LR} \\ \mathbf{A}_{RL} & \mathbf{A}_R \end{pmatrix} \cdot \begin{bmatrix} \mathbf{x}_L \\ \mathbf{x}_R \end{bmatrix} = \begin{bmatrix} \mathbf{b}_L \\ \mathbf{b}_R \end{bmatrix} \iff \begin{bmatrix} \mathbf{b}_L \\ \mathbf{b}_R \end{bmatrix} = \begin{bmatrix} \mathbf{A}_L \cdot \mathbf{x}_L + \mathbf{A}_{LR} \cdot \mathbf{x}_R \\ \mathbf{A}_{RL} \cdot \mathbf{x}_L + \mathbf{A}_R \cdot \mathbf{x}_R \end{bmatrix} \tag{2.23}$$

$$\mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(i)} = \mathbf{A} \cdot (\mathbf{x} - \mathbf{x}^{(i)}) = \begin{bmatrix} \mathbf{A}_L \cdot (\mathbf{x} - \mathbf{x}^{(i)})_L + \mathbf{A}_{LR} \cdot (\mathbf{x} - \mathbf{x}^{(i)})_R \\ \mathbf{A}_{RL} \cdot (\mathbf{x} - \mathbf{x}^{(i)})_L + \mathbf{A}_R \cdot (\mathbf{x} - \mathbf{x}^{(i)})_R \end{bmatrix} \tag{2.24}$$

In the following, we present some properties of the submatrix $\mathbf{A}_L$, that allow to formulate an hypothesis for the algebraic errors that suits the 2×2 block splitting.

Lemma 2.1 (L-Superiority wrt Ω_1)**.** *Let* $\mathbf{w}$ *be an arbitrary vector. The following inequality holds:*

$$\mathbf{w}_L^{\mathrm{T}} \cdot \mathbf{A}_L \cdot \mathbf{w}_L \geq \mathbf{w}_L^{\mathrm{T}} \cdot \mathbf{A}^{(1)} \cdot \mathbf{w}_L \ .$$

Proof. We know that $\mathbf{A}^{(2)}$ is a symmetric positive semi-definite (SPSD) matrix because it is the local stiffness matrix for subdomain Ω_2. Since $\mathbf{A}_{\mathrm{int}}^{(2)}$ is a principal submatrix of $\mathbf{A}^{(2)}$, it is SPSD as well. Therefore, the matrix $\mathbf{A}_L - \mathbf{A}^{(1)}$ from (2.22) is SPSD too and we have:

$$(\mathbf{w}_L^{\mathrm{T}} \cdot \mathbf{A}_L \cdot \mathbf{w}_L) - (\mathbf{w}_L^{\mathrm{T}} \cdot \mathbf{A}^{(1)} \cdot \mathbf{w}_L) = \mathbf{w}_L^{\mathrm{T}} \cdot (\mathbf{A}_L - \mathbf{A}^{(1)}) \cdot \mathbf{w}_L \geq 0 \ .$$

□

From this lemma, hypothesis (2.19) and the equality

$$(\mathbf{x} - \mathbf{x}^{(j)})^{\mathrm{T}} \cdot \mathbf{A}_p^{(1)} \cdot (\mathbf{x} - \mathbf{x}^{(j)}) = (\mathbf{x} - \mathbf{x}^{(j)})_L^{\mathrm{T}} \cdot \mathbf{A}^{(1)} \cdot (\mathbf{x} - \mathbf{x}^{(j)})_L \ ,$$

we can derive the subsequent corollary.

Corollary 2.1 (L-Dominance)**.** *Let* $\mathbf{x}^{(j)}$ *be an arbitrary vector for which* (2.19) *is satisfied. Then the following inequalities hold:*

$$(\mathbf{x} - \mathbf{x}^{(j)})_L^{\mathrm{T}} \cdot \mathbf{A}_L \cdot (\mathbf{x} - \mathbf{x}^{(j)})_L \geq (\mathbf{x} - \mathbf{x}^{(j)})^{\mathrm{T}} \cdot \mathbf{A}_p^{(1)} \cdot (\mathbf{x} - \mathbf{x}^{(j)}) \gg (\mathbf{x} - \mathbf{x}^{(j)})^{\mathrm{T}} \cdot \mathbf{A}_p^{(2)} \cdot (\mathbf{x} - \mathbf{x}^{(j)}).$$

If we rewrite the contribution of each set to the energy norm of the error, we have for any approximate solution $\mathbf{x}^{(i)}$ of the initial system (2.4):

$$||\mathbf{x} - \mathbf{x}^{(i)}||_{\mathbf{A}}^2 = \langle \mathbf{A} \cdot (\mathbf{x} - \mathbf{x}^{(i)}), \mathbf{x} - \mathbf{x}^{(i)} \rangle = \underbrace{\langle \mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(i)}, \mathbf{x} - \mathbf{x}^{(i)} \rangle_L}_{:= L\text{-term}} + \underbrace{\langle \mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(i)}, \mathbf{x} - \mathbf{x}^{(i)} \rangle_R}_{:= R\text{-term}}, \tag{2.25}$$

where

$$\begin{aligned} \langle \mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(i)}, \mathbf{x} - \mathbf{x}^{(i)} \rangle_L &= ||(\mathbf{x} - \mathbf{x}^{(i)})_L||_{\mathbf{A}_L}^2 + (\mathbf{x} - \mathbf{x}^{(i)})_L^{\mathrm{T}} \cdot \mathbf{A}_{LR} \cdot (\mathbf{x} - \mathbf{x}^{(i)})_R, \\ \langle \mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(i)}, \mathbf{x} - \mathbf{x}^{(i)} \rangle_R &= ||(\mathbf{x} - \mathbf{x}^{(i)})_R||_{\mathbf{A}_R}^2 + (\mathbf{x} - \mathbf{x}^{(i)})_R^{\mathrm{T}} \cdot \mathbf{A}_{RL} \cdot (\mathbf{x} - \mathbf{x}^{(i)})_L \ . \end{aligned} \tag{2.26}$$

Since $\mathbf{A}$ is symmetric, we have the equality of the coupling terms:

$$(\mathbf{x} - \mathbf{x}^{(i)})_R^{\mathrm{T}} \cdot \mathbf{A}_{RL} \cdot (\mathbf{x} - \mathbf{x}^{(i)})_L = (\mathbf{x} - \mathbf{x}^{(i)})_L^{\mathrm{T}} \cdot \mathbf{A}_{LR} \cdot (\mathbf{x} - \mathbf{x}^{(i)})_R \ .$$

As expressed in Corollary 2.1, a concentrated algebraic error on a subdomain Ω_1 implies that the $\mathbf{A}_L$-inner product of the error is dominant, and so will be the L-term, according to Equation (2.26). This is why they should be reduced for an efficient decrease of the energy norm of the error. We recognize that reducing the $\mathbf{A}_L$-inner product is a rather delicate matter, because the vectors $(\mathbf{x} - \mathbf{x}^{(i)})_L$ and $\mathbf{A}_L \cdot (\mathbf{x} - \mathbf{x}^{(i)})_L$ are unknown. The alternative that we propose and we consider reasonable is to take into account a coupling term as well, in order to retrieve a partial residual $(\mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(i)})_L$ that is computable. Then, we can expect that in (2.25), the L-term is dominant in the global energy norms from the i-th iteration when hypothesis (2.19) holds, and we seek a process to efficiently decrease them during the preconditioned solve.

2.5 Adaptive preconditioner for PCG based on local error indicators

Given that the linear systems dealt with are symmetric, we consider a PCG solver in the following. On the basis of the matrix decomposition described in 2.4.2, we introduce an adaptive preconditioning strategy enabling to reduce high local algebraic errors when solving the preconditioned linear system. The application of such a preconditioner starts after some few iterations that serve as an initialization phase, and is combined to a specific initial guess for the subsequent iterations of PCG solver. In the early stages of this study, our attention was clearly focused on substructuring methods that inspired us to apply to targeted error areas of the domain a similar treatment to the one foreseen for interface degrees of freedom in substructuring. The article [87] sets a reference framework for our study. It presents partitioned matrix methods along with Schur complement methods and establishes the equivalence between those two when PCG is used. It further suggests a more general form for the preconditioner where the local solves need not be carried out exactly.

2.5.1 Partitioned preconditioners suited for error reduction

As just explained above in Section 2.4.2, a good and affordable idea for ensuring the decay of local high algebraic errors seems to be to reduce the partial residual associated to the set of nodes L. In this respect, the techniques based on Schur complement allow to dissociate the L and R-parts. The following theorem due to Eisenstat (see [87] and references therein) states an equivalence between the Schur complement solve and a regular solve on the global system, with special initial guess and preconditioner. For the sequel, we denote the Schur complement matrix $\mathbf{S} := \mathbf{A}_R - \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR}$, and the modified right hand side $\mathbf{g} := \mathbf{b}_R - \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{b}_L$.

Theorem 2.1. *Using the same notation introduced above, let $\mathbf{x}_S^{(k)}$ be the k-th iterate of PCG solve of the system $\mathbf{S}\cdot\mathbf{x}_S = \mathbf{g}$ with initial guess $\mathbf{x}_S^{(0)}$ and preconditioner $\mathbf{M}_S$, and $\mathbf{x}^{(k)}$ be the k-th iterate of PCG solve of the system $\mathbf{A}\cdot\mathbf{x} = \mathbf{b}$ with initial guess $\mathbf{x}^{(0)}$ and preconditioner $\mathbf{M}$ such that:*

$$\mathbf{x}^{(0)} = \begin{bmatrix} \mathbf{A}_L^{-1}\cdot(\mathbf{b}_L - \mathbf{A}_{LR}\cdot\mathbf{x}_S^{(0)}) \\ \mathbf{x}_S^{(0)} \end{bmatrix} \; ; \; \mathbf{M} = \begin{pmatrix} \mathbf{A}_L & \mathbf{A}_{LR} \\ \mathbf{A}_{RL} & \mathbf{M}_S + \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR} \end{pmatrix}. \tag{2.27}$$

Then there holds, at each iteration k,

$$\mathbf{x}^{(k)} = \begin{bmatrix} \mathbf{A}_L^{-1}\cdot(\mathbf{b}_L - \mathbf{A}_{LR}\cdot\mathbf{x}_S^{(k)}) \\ \mathbf{x}_S^{(k)} \end{bmatrix}.$$

Proof. We give here an alternative proof to the one given in [87, Theorem 2.1 (i)] as we demonstrate by induction the results stated. We keep the notation used in [123, Algorithm 9.1,Chapter 9] for the PCG algorithm for solving (2.4) and we use an S subscript for the formulas associated to the system $\mathbf{S}\cdot\mathbf{x}_S = \mathbf{g}$. Let $\mathbf{x}^{(k)}$ (resp. $\mathbf{x}_S^{(k)}$) be the approximate solution iterates, $\mathbf{p}^{(k)}$ (resp. $\mathbf{p}_S^{(k)}$) the search directions, $\mathbf{r}^{(k)}$ (resp. $\mathbf{r}_S^{(k)}$) the residual iterates, $\mathbf{z}^{(k)}$ (resp. $\mathbf{z}_S^{(k)}$) the residuals for the preconditioned systems, and $\alpha^{(k)}$ (resp. $\alpha_S^{(k)}$) the step sizes for solving the global system (2.4) (resp.

the reduced system $\mathbf{S}\cdot\mathbf{x}_S=\mathbf{g}$) by PCG.

Note that the inverse of $\mathbf{M}$ can be expressed as:

$$\mathbf{M}^{-1}=\begin{pmatrix}\mathbf{A}_L^{-1}+\mathbf{A}_L^{-1}\mathbf{A}_{LR}\mathbf{M}_S^{-1}\mathbf{A}_{RL}\mathbf{A}_L^{-1} & -\mathbf{A}_L^{-1}\mathbf{A}_{LR}\mathbf{M}_S^{-1}\\ -\mathbf{M}_S^{-1}\mathbf{A}_{RL}\mathbf{A}_L^{-1} & \mathbf{M}_S^{-1}\end{pmatrix}.$$

Next, we proceed by induction on $k\in\mathbb{N}$ to prove that:

$$\mathbf{x}^{(k)}=\begin{bmatrix}\mathbf{A}_L^{-1}\cdot(\mathbf{b}_L-\mathbf{A}_{LR}\cdot\mathbf{x}_S^{(k)})\\ \mathbf{x}_S^{(k)}\end{bmatrix};\ \mathbf{r}^{(k)}=\begin{bmatrix}\mathbf{0}\\ \mathbf{r}_S^{(k)}\end{bmatrix};\ \mathbf{z}^{(k)}=\begin{bmatrix}-\mathbf{A}_L^{-1}\mathbf{A}_{LR}\cdot\mathbf{z}_S^{(k)}\\ \mathbf{z}_S^{(k)}\end{bmatrix}$$
$$\mathbf{p}^{(k)}=\begin{bmatrix}-\mathbf{A}_L^{-1}\mathbf{A}_{LR}\cdot\mathbf{p}_S^{(k)}\\ \mathbf{p}_S^{(k)}\end{bmatrix};\ (\mathbf{r}^{(k)})^{\mathrm{T}}\mathbf{z}^{(k)}=(\mathbf{r}_S^{(k)})^{\mathrm{T}}\mathbf{z}_S^{(k)};\ \alpha^{(k)}=\alpha_S^{(k)}.$$

For $k=0$: the first equality is satisfied by definition for $\mathbf{x}^{(0)}$. For the other equalities we have:

$$\begin{aligned}\mathbf{r}^{(0)}&=\mathbf{b}-\mathbf{A}\cdot\mathbf{x}^{(0)}=\begin{bmatrix}\mathbf{b}_L\\ \mathbf{b}_R\end{bmatrix}-\begin{bmatrix}\mathbf{b}_L-\mathbf{A}_{LR}\cdot\mathbf{x}_S^{(0)}+\mathbf{A}_{LR}\cdot\mathbf{x}_S^{(0)}\\ \mathbf{A}_{RL}\mathbf{A}_L^{-1}\cdot(\mathbf{b}_L-\mathbf{A}_{LR}\cdot\mathbf{x}_S^{(0)})+\mathbf{A}_R\cdot\mathbf{x}_S^{(0)}\end{bmatrix}\\ &=\begin{bmatrix}\mathbf{0}\\ \mathbf{g}-\mathbf{S}\cdot\mathbf{x}_S^{(0)}\end{bmatrix}=\begin{bmatrix}\mathbf{0}\\ \mathbf{r}_S^{(0)}\end{bmatrix}\\ \mathbf{z}^{(0)}&=\mathbf{M}^{-1}\cdot\mathbf{r}^{(0)}=\mathbf{M}^{-1}\cdot\begin{bmatrix}\mathbf{0}\\ \mathbf{r}_S^{(0)}\end{bmatrix}=\begin{bmatrix}-\mathbf{A}_L^{-1}\mathbf{A}_{LR}\mathbf{M}_S^{-1}\cdot\mathbf{r}_S^{(0)}\\ \mathbf{M}_S^{-1}\cdot\mathbf{r}_S^{(0)}\end{bmatrix}=\begin{bmatrix}-\mathbf{A}_L^{-1}\mathbf{A}_{LR}\cdot\mathbf{z}_S^{(0)}\\ \mathbf{z}_S^{(0)}\end{bmatrix}\\ \mathbf{p}^{(0)}&=\mathbf{z}^{(0)}=\begin{bmatrix}-\mathbf{A}_L^{-1}\mathbf{A}_{LR}\cdot\mathbf{z}_S^{(0)}\\ \mathbf{z}_S^{(0)}\end{bmatrix}=\begin{bmatrix}-\mathbf{A}_L^{-1}\mathbf{A}_{LR}\cdot\mathbf{p}_S^{(0)}\\ \mathbf{p}_S^{(0)}\end{bmatrix};\\ \mathbf{A}\cdot\mathbf{p}^{(0)}&=\begin{bmatrix}\mathbf{0}\\ -\mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR}\cdot\mathbf{p}_S^{(0)}+\mathbf{A}_R\cdot\mathbf{p}_S^{(0)}\end{bmatrix}=\begin{bmatrix}\mathbf{0}\\ \mathbf{S}\cdot\mathbf{p}_S^{(0)}\end{bmatrix}.\end{aligned}$$

Then, $\mathbf{p}^{(0)^{\mathrm{T}}}\mathbf{A}\cdot\mathbf{p}^{(0)}=\mathbf{p}_S^{(0)^{\mathrm{T}}}\mathbf{S}\cdot\mathbf{p}_S^{(0)}$.

$$\mathbf{r}^{(0)^{\mathrm{T}}}\mathbf{z}^{(0)}=\mathbf{r}_S^{(0)^{\mathrm{T}}}\mathbf{z}_S^{(0)};\ \text{ therefore }\ \alpha^{(0)}=\frac{\mathbf{r}^{(0)^{\mathrm{T}}}\mathbf{z}^{(0)}}{\mathbf{p}^{(0)^{\mathrm{T}}}\mathbf{A}\cdot\mathbf{p}^{(0)}}=\frac{\mathbf{r}_S^{(0)^{\mathrm{T}}}\mathbf{z}_S^{(0)}}{\mathbf{p}_S^{(0)^{\mathrm{T}}}\mathbf{S}\cdot\mathbf{p}_S^{(0)}}=\alpha_S^{(0)}.$$

Let $k \in \mathbb{N}$, we assume the equalities above are true for k, we then have:

$$\begin{aligned}
\mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{p}^{(k)} = \begin{bmatrix} \mathbf{A}_L^{-1}\cdot(\mathbf{b}_L - \mathbf{A}_{LR}\cdot\mathbf{x}_S^{(k)}) \\ \mathbf{x}_S^{(k)} \end{bmatrix} + \alpha_S^{(k)}\begin{bmatrix} -\mathbf{A}_L^{-1}\mathbf{A}_{LR}\cdot\mathbf{p}_S^{(k)} \\ \mathbf{p}_S^{(k)} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{A}_L^{-1}\cdot(\mathbf{b}_L - \mathbf{A}_{LR}\cdot(\mathbf{x}_S^{(k)} + \alpha_S^{(k)}\mathbf{p}_S^{(k)})) \\ \mathbf{x}_S^{(k)} + \alpha_S^{(k)}\mathbf{p}_S^{(k)} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_L^{-1}\cdot(\mathbf{b}_L - \mathbf{A}_{LR}\cdot\mathbf{x}_S^{(k+1)}) \\ \mathbf{x}_S^{(k+1)} \end{bmatrix} \\
\mathbf{r}^{(k+1)} &= \mathbf{r}^{(k)} - \alpha^{(k)}\mathbf{A}\cdot\mathbf{p}^{(k)} = \begin{bmatrix} \mathbf{0} \\ \mathbf{r}_S^{(k)} \end{bmatrix} - \alpha_S^{(k)}\begin{bmatrix} \mathbf{0} \\ -\mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR}\cdot\mathbf{p}_S^{(k)} + \mathbf{A}_R\cdot\mathbf{p}_S^{(k)} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{0} \\ \mathbf{r}_S^{(k)} - \alpha_S^{(k)}\mathbf{S}\cdot\mathbf{p}_S^{(k)} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{r}_S^{(k+1)} \end{bmatrix}; \\
\mathbf{z}^{(k+1)} &= \mathbf{M}^{-1}\cdot\mathbf{r}^{(k+1)} = \begin{bmatrix} -\mathbf{A}_L^{-1}\mathbf{A}_{LR}\mathbf{M}_S^{-1}\cdot\mathbf{r}_S^{(k+1)} \\ \mathbf{M}_S^{-1}\cdot\mathbf{r}_S^{(k+1)} \end{bmatrix} = \begin{bmatrix} -\mathbf{A}_L^{-1}\mathbf{A}_{LR}\mathbf{z}_S^{(k+1)} \\ \mathbf{z}_S^{(k+1)} \end{bmatrix}.
\end{aligned}$$

As a consequence, we have:

$$(\mathbf{r}^{(k+1)})^{\mathrm{T}}\mathbf{z}^{(k+1)} = (\mathbf{r}_S^{(k+1)})^{\mathrm{T}}\mathbf{z}_S^{(k+1)}$$

Combining this latter equality with the one stemming from the previous step, we obtain

$$\beta^{(k)} = \frac{(\mathbf{r}^{(k+1)})^{\mathrm{T}}\mathbf{z}^{(k+1)}}{(\mathbf{r}^{(k)})^{\mathrm{T}}\mathbf{z}^{(k)}} = \frac{(\mathbf{r}_S^{(k+1)})^{\mathrm{T}}\mathbf{z}_S^{(k+1)}}{(\mathbf{r}_S^{(k)})^{\mathrm{T}}\mathbf{z}_S^{(k)}} = \beta_S^{(k)}.$$

Thus,

$$\begin{aligned}
\mathbf{p}^{(k+1)} &= \begin{bmatrix} -\mathbf{A}_L^{-1}\mathbf{A}_{LR}\cdot(\mathbf{z}_S^{(k+1)} + \beta_S^{(k)}\mathbf{p}_S^{(k)}) \\ \mathbf{z}_S^{(k+1)} + \beta_S^{(k)}\mathbf{p}_S^{(k)} \end{bmatrix} = \begin{bmatrix} -\mathbf{A}_L^{-1}\mathbf{A}_{LR}\cdot\mathbf{p}_S^{(k+1)} \\ \mathbf{p}_S^{(k+1)} \end{bmatrix}; \\
\mathbf{A}\cdot\mathbf{p}^{(k+1)} &= \begin{bmatrix} \mathbf{0} \\ \mathbf{S}\cdot\mathbf{p}_S^{(k+1)} \end{bmatrix};
\end{aligned}$$

Finally $\alpha^{(k+1)} = \dfrac{\mathbf{r}^{(k+1)^{\mathrm{T}}}\mathbf{z}^{(k+1)}}{\mathbf{p}^{(k+1)^{\mathrm{T}}}\mathbf{A}\cdot\mathbf{p}^{(k+1)}} = \dfrac{\mathbf{r}_S^{(k+1)^{\mathrm{T}}}\mathbf{z}_S^{(k+1)}}{\mathbf{p}_S^{(k+1)^{\mathrm{T}}}\mathbf{S}\cdot\mathbf{p}_S^{(k+1)}} = \alpha_S^{(k+1)}.$ □

In the remainder of this section, we generalize Theorem 2.1 to provide sufficient and necessary conditions on the initial guess and the preconditioner to obtain a nil residual on L at each iteration.

Theorem 2.2 (Sufficient condition for nil residual on L-part)**.** *We denote* $\mathbf{W}$ *the Cholesky factor of* $\mathbf{M}$*:* $\mathbf{M} = \mathbf{W}\mathbf{W}^{\mathrm{T}}$*,* n_L *and* n_R *designate the sizes of the diagonal blocks* $\mathbf{A}_L$ *and* $\mathbf{A}_R$ *respectively. Let* $\mathbf{x}_R^{(0)}$ *be an arbitrary vector of length* n_R *and* $\mathbf{W}_1$*,* $\mathbf{W}_2$ *two invertible matrices of sizes* n_L *and* n_R *respectively. Let the linear system* (2.4) *be solved by a PCG solver with a preconditioner* $\mathbf{M} = \mathbf{W}\mathbf{W}^{\mathrm{T}}$ *and an initial guess* $\mathbf{x}^{(0)}$ *such that:*

$$\mathbf{x}^{(0)} = \begin{bmatrix} \mathbf{A}_L^{-1}\cdot(\mathbf{b}_L - \mathbf{A}_{LR}\cdot\mathbf{x}_R^{(0)}) \\ \mathbf{x}_R^{(0)} \end{bmatrix}, \quad \mathbf{W} = \begin{pmatrix} \mathbf{W}_1 & \mathbf{0} \\ \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{W}_1 & \mathbf{W}_2 \end{pmatrix}; \tag{2.28}$$

then $(\mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(k)})_L = \mathbf{0}$ *at each iteration k of PCG.*

Proof. With

$$\mathbf{x}^{(0)} = \begin{bmatrix} \mathbf{A}_L^{-1} \cdot (\mathbf{b}_L - \mathbf{A}_{LR} \cdot \mathbf{x}_R^{(0)}) \\ \mathbf{x}_R^{(0)} \end{bmatrix},$$

there holds:

$$\mathbf{r}^{(0)} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b}_R - \mathbf{A}_{RL}\mathbf{A}_L^{-1} \cdot \mathbf{b}_L + (\mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR} - \mathbf{A}_R) \cdot \mathbf{x}_R^{(0)} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{g} - \mathbf{S} \cdot \mathbf{x}_R^{(0)} \end{bmatrix}.$$

Therefore

$$\mathbf{W}^{-1} \cdot \mathbf{r}^{(0)} = \begin{bmatrix} \mathbf{0} \\ \mathbf{W}_2^{-1} \cdot (\mathbf{g} - \mathbf{S} \cdot \mathbf{x}_R^{(0)}) \end{bmatrix}.$$

Besides, for each iteration k there exists a polynomial q_k of degree k such that:

$$\mathbf{W}^{-1} \cdot \mathbf{r}^{(k)} = q_k(\mathbf{W}^{-1}\mathbf{A}\mathbf{W}^{-\mathrm{T}})\mathbf{W}^{-1} \cdot \mathbf{r}^{(0)} \tag{2.29}$$

The definition of preconditioner $\mathbf{W}$ in (4.27) yields:

$$\mathbf{W}^{-1}\mathbf{A}\mathbf{W}^{-\mathrm{T}} = \begin{pmatrix} \mathbf{W}_1^{-1}\mathbf{A}_L\mathbf{W}_1^{-\mathrm{T}} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_2^{-1}\mathbf{S}\mathbf{W}_2^{-\mathrm{T}} \end{pmatrix} \tag{2.30}$$

Then

$$q_k(\mathbf{W}^{-1}\mathbf{A}\mathbf{W}^{-\mathrm{T}}) = \begin{pmatrix} q_k(\mathbf{W}_1^{-1}\mathbf{A}_L\mathbf{W}_1^{-\mathrm{T}}) & \mathbf{0} \\ \mathbf{0} & q_k(\mathbf{W}_2^{-1}\mathbf{S}\mathbf{W}_2^{-\mathrm{T}}) \end{pmatrix}$$

Consequently, from (2.29) we deduce that:

$$\mathbf{r}^{(k)} = \begin{bmatrix} \mathbf{0} \\ \mathbf{W}_2 q_k(\mathbf{W}_2^{-1}\mathbf{S}\mathbf{W}_2^{-\mathrm{T}})\mathbf{W}_2^{-1} \cdot (\mathbf{g} - \mathbf{S} \cdot \mathbf{x}_R^{(0)}) \end{bmatrix}$$

□

Remark 2.5. When the sufficient condition above is satisfied, the preconditioner $\mathbf{M}$ has the following shape:

$$\begin{aligned} \mathbf{M} &= \begin{pmatrix} \mathbf{W}_1 & \mathbf{0} \\ \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{W}_1 & \mathbf{W}_2 \end{pmatrix} \begin{pmatrix} \mathbf{W}_1^{\mathrm{T}} & \mathbf{W}_1^{\mathrm{T}}\mathbf{A}_L^{-\mathrm{T}}\mathbf{A}_{LR} \\ \mathbf{0} & \mathbf{W}_2^{\mathrm{T}} \end{pmatrix} \\ &= \left(\begin{array}{c|c} \mathbf{W}_1\mathbf{W}_1^{\mathrm{T}} & \mathbf{W}_1\mathbf{W}_1^{\mathrm{T}}\mathbf{A}_L^{-\mathrm{T}}\mathbf{A}_{LR} \\ \hline \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{W}_1\mathbf{W}_1^{\mathrm{T}} & \mathbf{W}_2\mathbf{W}_2^{\mathrm{T}} + \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{W}_1\mathbf{W}_1^{\mathrm{T}}\mathbf{A}_L^{-\mathrm{T}}\mathbf{A}_{LR} \end{array} \right) \end{aligned}$$

If we denote the two SPD matrices $\mathbf{M}_1 := \mathbf{W}_1\mathbf{W}_1^{\mathrm{T}}$ and $\mathbf{M}_2 := \mathbf{W}_2\mathbf{W}_2^{\mathrm{T}}$ then:

$$\mathbf{M} = \left(\begin{array}{c|c} \mathbf{M}_1 & \mathbf{M}_1\mathbf{A}_L^{-\mathrm{T}}\mathbf{A}_{LR} \\ \hline \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{M}_1 & \mathbf{M}_2 + \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{M}_1\mathbf{A}_L^{-\mathrm{T}}\mathbf{A}_{LR} \end{array} \right);$$

which is a generalization of the preconditioner defined in (2.27) of Theorem 2.1.

When the conditions of Theorem 2.2 are fulfilled, we can state that in addition to the **A**-orthogonality property, the residual vanishes on L at each iteration. The question that arises next is to know if the preconditioner defined in Theorem 2.2 is the only one that has this particular property or if there exist others.

Theorem 2.3 (Necessary condition for nil residual on L-part)**.** *Let* $\mathbf{M}$ *be a preconditioner of* $\mathbf{A}$ *such that the solve of* (2.4) *by PCG yields a vanishing residual on* L *at each iteration:*

$$\mathbf{r}_L^{(k)} = \mathbf{b}_L - \mathbf{A}_L \cdot \mathbf{x}_L^{(k)} - \mathbf{A}_{LR} \cdot \mathbf{x}_L^{(k)} = 0 \,, \qquad \forall k \leq \tilde{k} \quad ;$$

where $\tilde{k}$ *is the iteration when* $\mathbf{x}^{(\tilde{k})} = \mathbf{x}$*. Then we have:*

$$\dim(\mathrm{Ker}((\mathbf{A}\mathbf{M}^{-1})_{LR})) \geq \tilde{k} - 1 \,. \tag{2.31}$$

Proof. Still with the notation used in [123, Algorithm 9.1,Chapter 9] for the PCG algorithm, the successive residuals satisfy a two-term recurrence:

$$\mathbf{r}^{(i+1)} = -\alpha^{(i)}\mathbf{A}\mathbf{M}^{-1}\cdot\mathbf{r}^{(i)} + (1 + \frac{\alpha^{(i)}\beta^{(i-1)}}{\alpha^{(i-1)}})\mathbf{r}^{(i)} - \frac{\alpha^{(i)}\beta^{(i-1)}}{\alpha^{(i-1)}}\mathbf{r}^{(i-1)}, \forall i \in \mathbb{N}\backslash\{0\} \tag{2.32}$$

Let $i \in [\![1, \tilde{k}]\!]$ be arbitrary. A vanishing residual on L implies that

$$\mathbf{r}_L^{(i+1)} = \mathbf{r}_L^{(i)} = \mathbf{r}_L^{(i-1)} = 0 \,,$$

then due to (2.32), we get

$$(\mathbf{A}\mathbf{M}^{-1} \cdot \mathbf{r}^{(i)})_L = 0 \,;$$

which using the 2×2 block splitting for $\mathbf{A}\mathbf{M}^{-1}$ as in (2.20) and the fact that $\mathbf{r}_L^{(i)} = 0$ gives

$$(\mathbf{A}\mathbf{M}^{-1})_{LR} \cdot \mathbf{r}_R^{(i)} = 0 \tag{2.33}$$

We know that the vectors $(\mathbf{x} - \mathbf{x}^{(0)}, \ldots, \mathbf{x} - \mathbf{x}^{(k)})$ are linearly independent for every $k < \tilde{k}$. Because $\mathbf{A}$ is nonsingular, $(\mathbf{r}^{(0)} = \mathbf{A} \cdot (\mathbf{x} - \mathbf{x}^{(0)}), \ldots, \mathbf{r}^{(k)} = \mathbf{A} \cdot (\mathbf{x} - \mathbf{x}^{(k)}))$ are linearly independent as well. Since $\mathbf{r}_L^{(j)} = 0$ for $j = 0, \ldots, k$, we deduce that $(\mathbf{r}_R^{(0)}, \ldots, \mathbf{r}_R^{(k)})$ are also linearly independent for $k < \tilde{k}$.
According to (2.33), this implies that $\dim(\mathrm{Ker}((\mathbf{A}\mathbf{M}^{-1})_{LR})) \geq k \,, \forall k < \tilde{k}$
i.e. $\dim(\mathrm{Ker}((\mathbf{A}\mathbf{M}^{-1})_{LR})) \geq \tilde{k} - 1$. □

Theorems 2.2 and 2.3 provide respectively sufficient and necessary conditions on the preconditioner to obtain a nil residual on L at each iteration of PCG. There is a special case when these conditions match each other and the proposed preconditioner is unique. Indeed, for $\tilde{k} > n_R$, we have $\dim(\mathrm{Ker}((\mathbf{A}\mathbf{M}^{-1})_{LR})) \geq n_R$ and thus according to the rank-nullity theorem:

$$\mathrm{rank}((\mathbf{A}\mathbf{M}^{-1})_{LR}) = n_R - \dim(\mathrm{Ker}((\mathbf{A}\mathbf{M}^{-1})_{LR})) \leq 0.$$

Which means that $(\mathbf{A}\mathbf{M}^{-1})_{LR} = 0$. This latter equality is equivalent to the block diagonal shape (2.30) in the proof of Theorem 2.2.

2.5.2 Condition number improvement

In this subsection, we discuss the outcome of using the partitioned preconditioner of Theorem 2.1 with respect to a 2×2 block diagonal preconditioner in terms of a possible reduction of the condition number of the preconditioned operator. Considering $\mathbf{M}$ (resp. $\mathbf{M}_S$) the SPD preconditioner of $\mathbf{A}$ (resp. $\mathbf{S}$), we denote the following condition numbers:

$$\mathcal{K}(\mathbf{A}, \mathbf{M}) := \mathcal{K}(\mathbf{M}^{-\frac{1}{2}}\mathbf{A}\mathbf{M}^{-\frac{1}{2}}) \; ;$$
$$\mathcal{K}(\mathbf{S}, \mathbf{M}_S) := \mathcal{K}(\mathbf{M}_S^{-\frac{1}{2}}\mathbf{S}\mathbf{M}_S^{-\frac{1}{2}}) \; .$$

According to [103, Theorem 4.2.], for every block-diagonal preconditioner $\widetilde{\mathbf{M}}$ of $\mathbf{A}$ which has the shape:

$$\widetilde{\mathbf{M}} = \left(\begin{array}{c|c} \mathbf{A}_L & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{M}_R \end{array}\right) \tag{2.34}$$

such that $\mathbf{M}_R$ is an arbitrary preconditioner associated with the second diagonal block $\mathbf{A}_R$, we have:

$$\mathcal{K}(\mathbf{S}, \mathbf{M}_R) \leq \mathcal{K}(\mathbf{A}, \widetilde{\mathbf{M}})$$

Therefore, if we choose $\mathbf{M}_S = \mathbf{M}_R$, we still have:

$$\mathcal{K}(\mathbf{S}, \mathbf{M}_S) \leq \mathcal{K}(\mathbf{A}, \widetilde{\mathbf{M}}) \tag{2.35}$$

Consequently, since the convergence of iterative solvers is commonly affected by the conditioning of the matrix, from (2.35) we can expect that PCG for the Schur complement system converges faster than for the original system. According to Theorem 2.1, this amounts to saying that PCG on (2.4) converges faster when preconditioned by $\widehat{\mathbf{M}} = \begin{pmatrix} \mathbf{A}_L & \mathbf{A}_{LR} \\ \mathbf{A}_{RL} & \mathbf{M}_R + \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR} \end{pmatrix}$ than by the block-diagonal $\widetilde{\mathbf{M}} = \begin{pmatrix} \mathbf{A}_L & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_R \end{pmatrix}$. Later, in Section 2.6 where we show the numerical results, we observe this convergence improvement in practice.

2.5.3 Context of use

Let us suggest the context of the solution process for practical use:

❶ A PCG solver with a given preconditioner $\overline{\mathbf{M}}$ is run to solve (2.4).

❷ At an iteration step j_0, an intermediate solution $\mathbf{x}^{(j_0)}$ is computed to get an estimated algebraic error distribution.

↪ Local algebraic error $\eta_{\text{alg},K}^{(j_0)}$ is evaluated over each mesh elements $K \in \mathcal{T}_h$.

❸ This allows for marking the elements with largest errors and extracting their associated node indices.

↪ This step yields subdomain Ω_1 of (2.7) and (2.8), and L-subset.

❹ Proceed with permuting the system to obtain a L/R splitting as in (2.20).

❺ Perform an exact Cholesky factorization on the L-block, build the adaptive preconditioner and use $\mathbf{x}^{(j_0)}$ to compute a special initial guess.

↪ $\mathbf{W}_1$(such that $\mathbf{W}_1\mathbf{W}_1^{\mathrm{T}}=\mathbf{A}_L$), $\mathbf{x}^{(0)}$ and $\mathbf{M}$ as defined in (2.27) with $\mathbf{x}_S^{(0)}:=\mathbf{x}_R^{(j_0)}$.

❻ Run PCG on the permuted system with preconditioner $\mathbf{M}$ and the newly computed starting guess $\mathbf{x}^{(0)}$.

Remark 2.6. Concerning step n°5 above, for computing $\mathbf{M}$ in (2.27), the $\mathbf{M}_S$ matrix should ideally approximate the Schur complement matrix $\mathbf{S}$. Though there already exists a wide range of Schur complement methods [126, 32, 97], we suggest to recycle the preconditioner $\overline{\mathbf{M}}$ by extracting its R-block and using it as $\mathbf{M}_S$. This allows to save the time required to construct $\mathbf{M}$.

For the sake of comparison, we denote by process 2 the solve procedure described above, and by process 1 the solve with preconditioner $\overline{\mathbf{M}}$ on the initial system pursued to the end. We can say that the cost should differ between the two processes. In process 2, more effort is put into building the preconditioner. The additional cost compared to process 1 is the one related to the Cholesky factorization of $\mathbf{A}_L$. However, the cost of the solve is hopefully diminished due to the convergence improvement with $\mathbf{M}$ over $\overline{\mathbf{M}}$.

2.6 Numerical results

In this section, we choose different 2D elliptic problems as they usually serve as test cases for PDE solution algorithms. The following numerical experiments are based on Matlab with PDE toolbox in order to create a mesh and solve the considered PDE on a domain Ω.
We consider a triangular mesh. If the exact solution is known, the mesh is adaptively refined according to the distribution of the discretization error during an initialization step before running tests. Otherwise, the mesh in Delaunay, generated by Matlab initmesh command with the maximum element size specified by the parameter $H_{\max}$. We will call such mesh "uniform" in the sequel. Once the main linear system is defined, we run few iterations of the linear solver (20 iterations of PCG) to get a starting distribution of the a posteriori algebraic error estimates on all the elements of the mesh. From these quantities, we distinguish the L and R subdomains. Multiple strategies are conceivable for selecting the elements that will form our L-subdomain. We will define some of them in Section 2.6.1. Next, we solve by the procedure described in Section 2.5.3. Finally, we compare the evolution of the global energy norm and the L-norm of the error (that we define in Section 2.6.1) within iterations for process 1 and process 2.

2.6.1 Some strategies for initiating the adaptive procedure

In order to build the L-subdomain, we start by sorting the mesh elements according to the a posteriori algebraic error estimates per element. Then in order to select elements that will compose L, we need a certain threshold. One option is to take a ratio on the number of elements. It consists in setting a certain percentage "*perc*" on the total number of elements, and gathering the first *perc* % elements where the a posteriori algebraic error estimate is the largest. However, due to the complex distribution of errors over elements for some test cases, we believe it would be a

good idea to adjust the way we choose elements that form Ω_1. Henceforth, we apply the so-called Dörfler criterion [49]. It aims at finding the minimal set $\mathcal{E}_1$ within the set of all elements $\mathcal{E}$ such that for some parameter $\Theta \in]0,1[$:

$$L\text{-norm}^2 := \sum_{K\in\mathcal{E}_1} (\eta_{\text{alg},K})^2 \geq \Theta \left(\sum_{K\in\mathcal{E}} (\eta_{\text{alg},K})^2 \right) , \tag{2.36}$$

where $\eta_{\text{alg},K}$ denotes the a posteriori algebraic error estimate over the element K, and the term "L-norm" in this article will refer, somewhat imprecisely, to the portion of error captured in $\Omega_1 = \bigcup_{K\in\mathcal{E}_1} K$. Note that it is actually the $\mathbf{A}_p^{(1)}$-seminorm of the error:

$$L\text{-norm}^2 = \langle \mathbf{A}_p^{(1)} \cdot (\mathbf{x} - \mathbf{x}^{(j_0)}), \mathbf{x} - \mathbf{x}^{(j_0)} \rangle .$$

In our framework, we believe that (2.36) better reflects Hypothesis (2.8). Indeed, the advantage over the previous marking strategy is that we are selecting elements that concentrate a certain percentage of the error instead of directly choosing a percentage of the total number of elements.

Remark 2.7. Whenever the a posteriori discretization error estimates are known, we could exploit them as an indicator of the right iteration to estimate the starting a posteriori algebraic error distribution. Indeed, the evaluation of errors would initiate as soon as the iterative solve reaches an iteration j_0 such that:

$$\eta_{\text{alg}}^{(j_0)} \leq \gamma \, \eta_{\text{disc}}^{(j_0)} ,$$

where $\eta_{\text{alg}}^{(j_0)}$ and $\eta_{\text{disc}}^{(j_0)}$ are the total a posteriori algebraic and discretization error estimates respectively, $\gamma > 0$ is a scalar parameter. We expect that this approach would give reliable information on the error distribution since inequalities of this shape have been used as stopping criterion in many adaptive algorithms in the literature; see, e.g., [46, 55] and the references given there.

For the numerical experiments presented in this section, we decide to consider:

- a fixed value (20) for j_0,
- a Block-Jacobi preconditioner composed of 50 blocks for $\overline{\mathbf{M}}$,
- a stopping threshold value of 10^{-6} for the euclidean norm of the residual.

In the following, whenever the initial a posteriori algebraic error estimates' distribution is plotted, thick horizontal lines labeled with θ_1 and θ_2 on the color bar indicates the extent of the errors' range covered with the Dörfler rates Θ_1 and Θ_2 considered, i.e. all elements represented in color shades above the corresponding thresholds θ_1 and θ_2 respectively form Ω_1. As far as the convergence curves are concerned, the blue one corresponds to process 1 (PCG with block Jacobi preconditioner $\overline{\mathbf{M}}$) while process 2 (PCG with our proposed preconditioner) is represented in red and black. Detailed information about test configuration (mesh, Dörfler rate Θ (as %), size percentage (n_L in % of n)) and results (number of iterations for regular and adaptive solve procedures: it_{st}, it_{ada}) are given in Tables 2.1 and 2.2.

2.6.2 Poisson's equation

First, we focus exclusively on Poisson equations of the form

$$-\Delta \underline{u} = -\frac{\partial^2 \underline{u}}{\partial x^2} - \frac{\partial^2 \underline{u}}{\partial y^2} = \underline{f}(x,y) \quad \text{in} \quad \Omega \tag{2.37}$$

with homogeneous Dirichlet boundary condition

$$\underline{u} = 0 \quad \text{on} \quad \partial\Omega. \tag{2.38}$$

This is a particular case of problem (3.51) with the diffusion tensor equal to identity. For our test cases, we consider two classic examples with given smooth solutions $\underline{u}$ on the square $[-1,1]$:

$$\underline{u}^{(1)} = (x+1)\times(x-1)\times(y+1)\times(y-1)\times\exp(-\alpha\times(x^2+y^2))\ ; \tag{2.39}$$

$$\underline{u}^{(2)} = (x+1)\times(x-1)\times(y+1)\times(y-1)\times(\exp(-\alpha\times((x+0.5)^2 + (y+0.5)^2)) - \exp(-\beta\times((x-0.5)^2+(y-0.5)^2)))\ ; \tag{2.40}$$

with $\alpha = 4\,000$ and $\beta = 3\,000$.

As far as the mesh configuration is concerned, we consider two uniform meshes $\mathcal{M}^{(1)}$ and $\mathcal{M}^{(2)}$ with maximum edge sizes $H_{\max} = 0.1$ and $H_{\max} = 0.05$ respectively. Then, the total numbers of elements equal to 87 552 and 354 304 respectively. After discretization, the sizes of matrix $\mathbf{A}$ are $43\,457 \times 43\,457$ and $176\,513 \times 176\,513$ respectively.

For the first test case, the Galerkin solution is plotted in Figure 2.3, the initial distribution and a

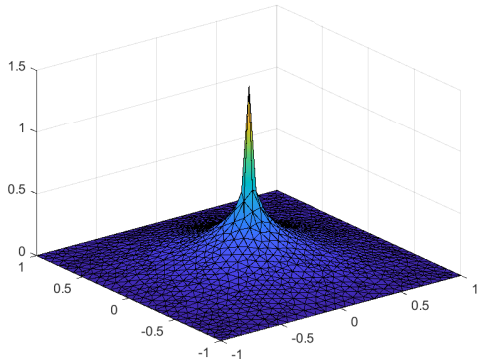
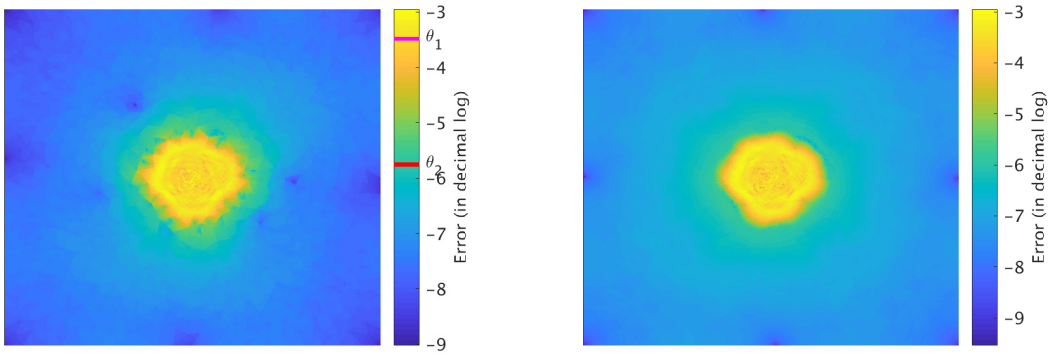


Figure 2.3 – *Galerkin solution* $u_h^{(1)}$

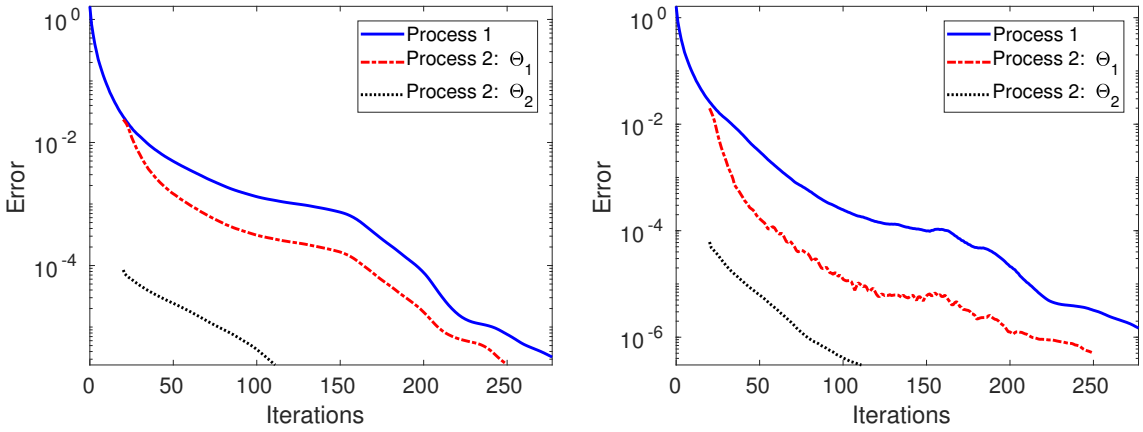
posteriori estimation of algebraic error (after $j_0 = 20$ iterations) over the domain Ω are shown in Figure 2.4 for the first mesh $\mathcal{M}^{(1)}$ and in Figure 2.6 for the second mesh $\mathcal{M}^{(2)}$; while the global energy norm and the L-norm of the error within iterations are shown in Figures 2.5 (for $\mathcal{M}^{(1)}$) and 2.7 (for $\mathcal{M}^{(2)}$) with two different values of the parameter Θ in the Dörfler criterion.



(a) Algebraic a posteriori error estimates after 20 iterations

(b) Algebraic errors after 20 iterations

Figure 2.4 – *Initial distribution and a posteriori estimation of algebraic error for test case n°1 on mesh* $\mathcal{M}^{(1)}$



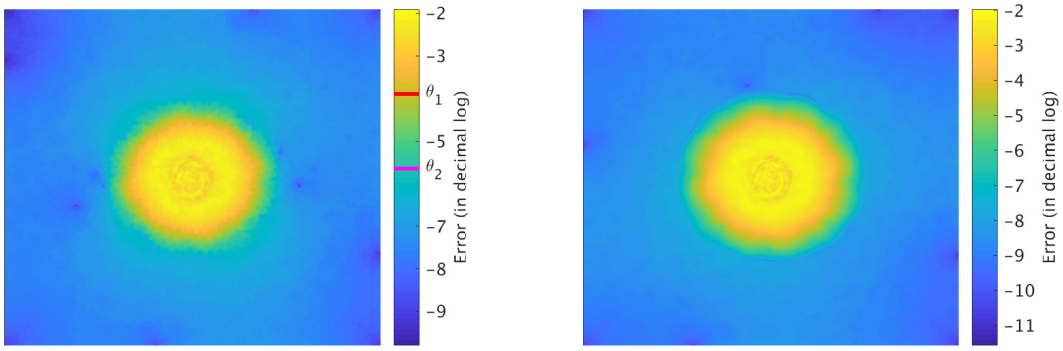
(a) Energy norm of the algebraic error

(b) L-norm of the algebraic error

Figure 2.5 – *Error evolution for test case n°1 on mesh* $\mathcal{M}^{(1)}$

With the first Dörfler rate Θ_1, Ω_1 does not take into account all the area where important errors are observed. We notice in Figures 2.5 and 2.7 the decrease of the global energy norm of the error and that of the algebraic error on L-marked elements is observed on both processes but more markedly for the process 2. Now, when we increase the Dörfler rate to a value Θ_2 close to one to cover almost all the high errors' region, we notice the rapid decrease of the global energy norm of the error and that of the algebraic error on L-marked elements with process 2 (black curve). The convergence is faster and there is a 3x and a major 14x speedups in terms of iterations on meshes $\mathcal{M}^{(1)}$ and $\mathcal{M}^{(2)}$ respectively (see Table 2.1).

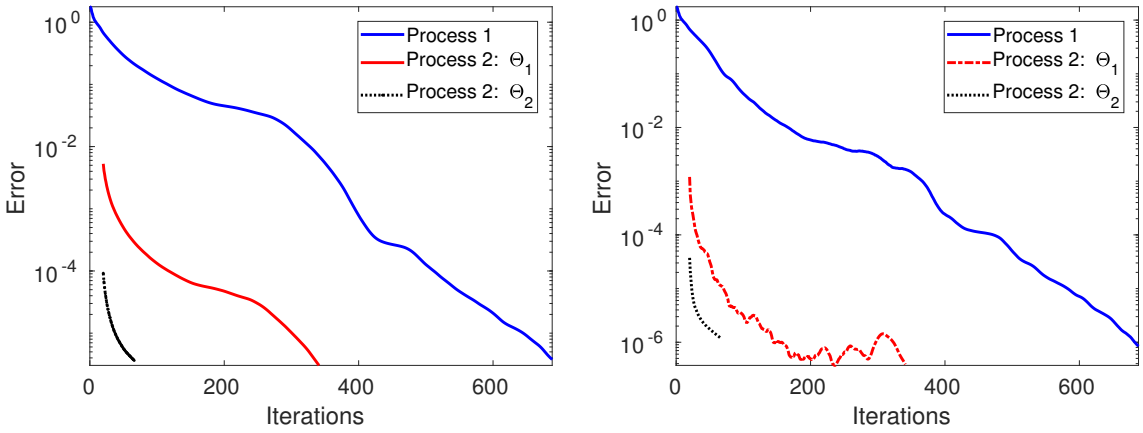
Likewise, we present the results obtained for the second test case. The Galerkin solution is plotted in Figure 2.8. Figures 2.9 and 2.11 display the initial error distribution over meshes $\mathcal{M}^{(1)}$ and $\mathcal{M}^{(2)}$. Figures 2.10 and 2.12 depict the evolution of the global energy norm and the L-norm of the error for $\mathcal{M}^{(1)}$ and $\mathcal{M}^{(2)}$ with two different values of the parameter Θ in the Dörfler criterion. In



(a) Algebraic a posteriori error estimates after 20 iterations

(b) Algebraic errors after 20 iterations

Figure 2.6 – *Initial distribution and a posteriori estimation of algebraic error for test case n°1 on mesh* $\mathcal{M}^{(2)}$



(a) Energy norm of the algebraic error

(b) L-norm of the algebraic error

Figure 2.7 – *Error evolution for test case n°1 on mesh* $\mathcal{M}^{(2)}$

those latter figures, we point out that the error reduction in process 2 with Θ_1 is better than in process 1. It becomes even more efficient with the larger Θ_2 as it yields a convergence after 65 or 11 iterations only according to the mesh resolution (see Table 2.2). That represents an important 5x (resp. 56x) speedup over PCG. With these results, one can see the potential that the adaptive method offers when the errors are localized (even on separate zones).

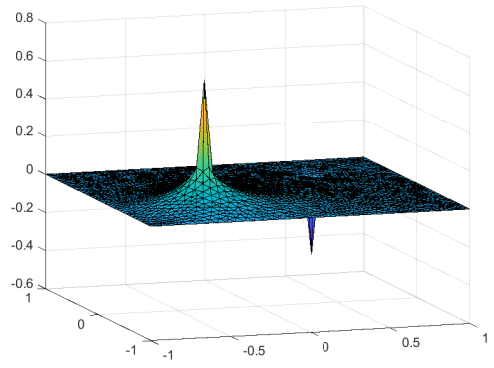
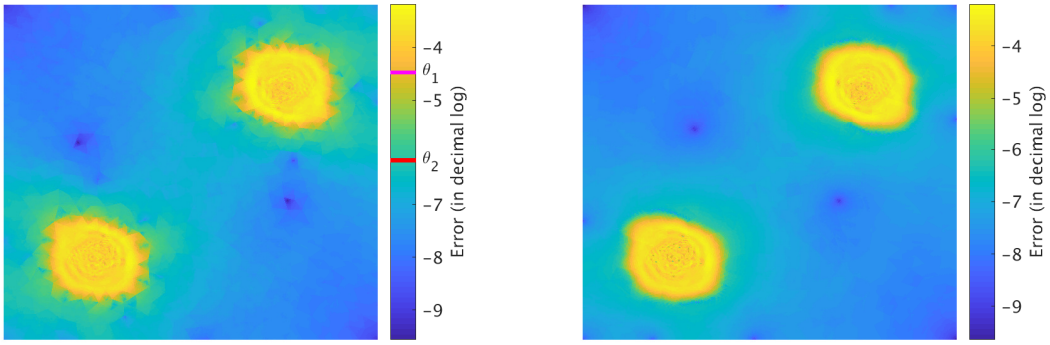


Figure 2.8 – *Galerkin solution* $u_h^{(2)}$



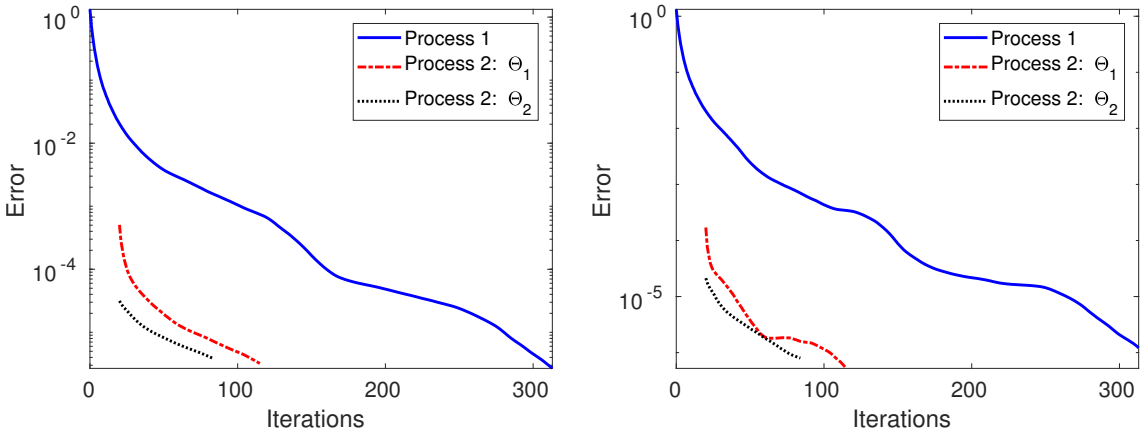
(a) Algebraic a posteriori error estimates after 20 iterations

(b) Algebraic errors after 20 iterations

Figure 2.9 – *Initial distribution and a posteriori estimation of algebraic error for test case n°2 on mesh* $\mathcal{M}^{(1)}$

Table 2.1 – *Test configuration and number of iterations for standard and adaptive processes with different values of the Dörfler rate* Θ *applied to Poisson problems.*

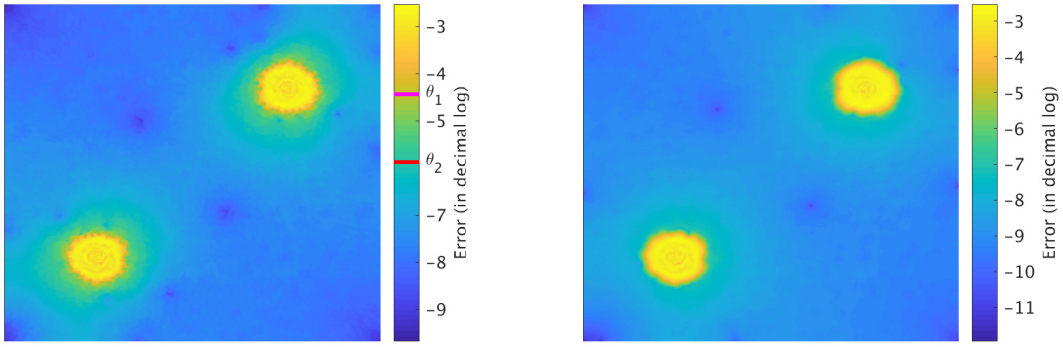
Configuration			**Iterations**		
Test case	Mesh	Θ (as %)	n_L in % of n	it_{st}	it_{ada}
1	$\mathcal{M}^{(1)}$	87.70	3.50	278	230
		99.99	10.99	278	92
1	$\mathcal{M}^{(2)}$	99	9.99	689	322
		99.99	15	689	47
2	$\mathcal{M}^{(1)}$	99.82	10	314	96
		99.99	25.17	314	65
2	$\mathcal{M}^{(2)}$	99	3.99	614	57
		99.99	10	614	11



(a) Energy norm of the algebraic error

(b) L-norm of the algebraic error

Figure 2.10 – *Error evolution for test case n°2 on mesh* $\mathcal{M}^{(1)}$



(a) Algebraic a posteriori error estimates after 20 iterations

(b) Algebraic errors after 20 iterations

Figure 2.11 – *Initial distribution and a posteriori estimation of algebraic error for test case n°2 on mesh* $\mathcal{M}^{(2)}$

2.6.3 Diffusion equation with inhomogeneous coefficient

In this section, we tackle diffusion problems with inhomogeneous diffusion tensor of the form

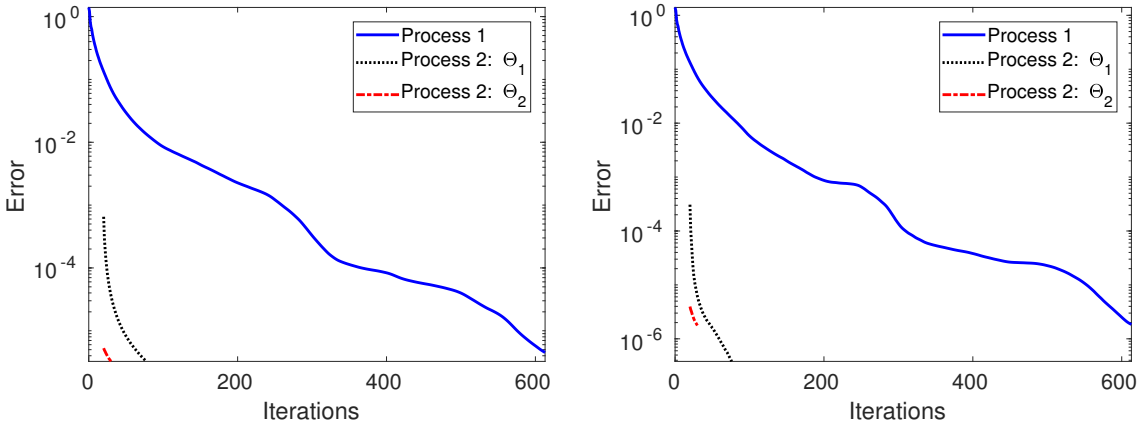
$$-\nabla\cdot(\underline{\mathbf{K}}\nabla\underline{u}) = \underline{f}(x,y) \quad \text{in} \quad \Omega =]0,1[\tag{2.41}$$

with Dirichlet boundary condition

$$\underline{u} = \underline{u}_0 \quad \text{on} \quad \partial\Omega \tag{2.42}$$

This time, we define the right hand side $\underline{f}$ as the constant function taking the value 1 on Ω. The Dirichlet boundary condition is prescribed on $\partial\Omega$ by the function:

$$\underline{u}_0(x,y) = \sqrt{x}; \tag{2.43}$$



(a) Energy norm of the algebraic error (b) L-norm of the algebraic error

Figure 2.12 – *Error evolution for test case n°2 on mesh* $\mathcal{M}^{(2)}$

In the diffusion part of this numerical section, we have considered the algebraic error to assess the validity of the adaptive procedure when the error detection is ideal (optimal case).

The diffusivity taken here is a highly heterogeneous function of Ω. In the sequel, we will consider two configurations of the diffusivity. In both cases, the diffusion tensor is defined as a multiple of the identity matrix: $\underline{\mathbf{K}} = c * \mathbf{I}$; and the multiplication factor c varies through the domain Ω. In the first test, the diffusivity is defined as in [80, Section 5]:

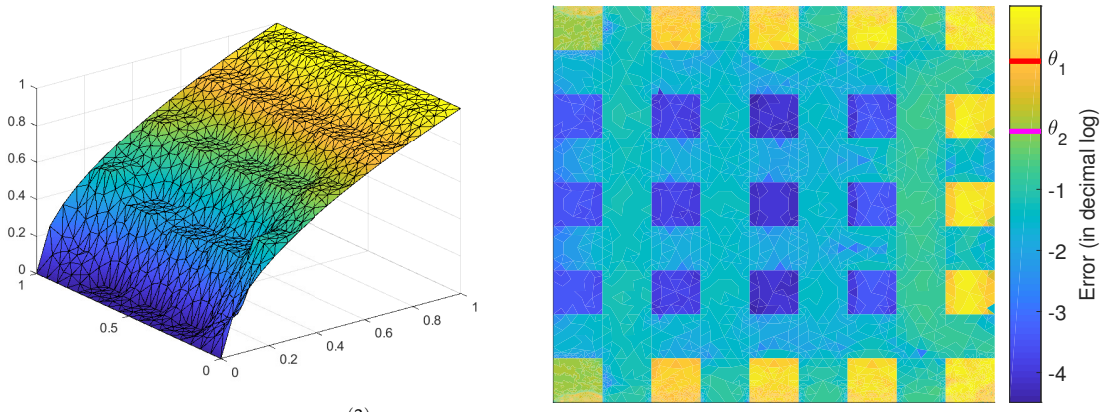
$$c^{(3)}(x,y) = \begin{cases} 10^5(\lfloor 9x \rfloor + 1) & \text{if} \quad \lfloor (9x) \rfloor \equiv 0 \pmod 2 \text{ and } \lfloor (9y) \rfloor \equiv 0 \pmod 2, \\ 1 & \text{otherwise.} \end{cases}$$

We consider a uniform mesh with a maximum edge size $H_{\max} = 0.03$ and 30 257 mesh elements. After discretization, the size of the matrix $\mathbf{A}$ is $14\,690 \times 14\,690$. The Galerkin solution and the initial algebraic error distribution over the domain Ω are shown in Figure 2.13, while the global energy norm and the L-norm of the error within iterations are plotted in Figure 2.14. We observe that curves of process 2 are almost below the curve of process 1 for the L-norm in Figure 2.14. We also highlight that by going from $\Theta_1 = 0.95$ to $\Theta_2 = 0.99$ to take into account more error zones, we increase the size of $\mathbf{A}_L$ by a factor of three, but the reduction in number of iterations with respect to PCG was five times larger (8% $\rightarrow$ 41%).

For the next test case, we consider the second diffusivity shown in Figure 2.15. The formula of the corresponding diffusivity function is:

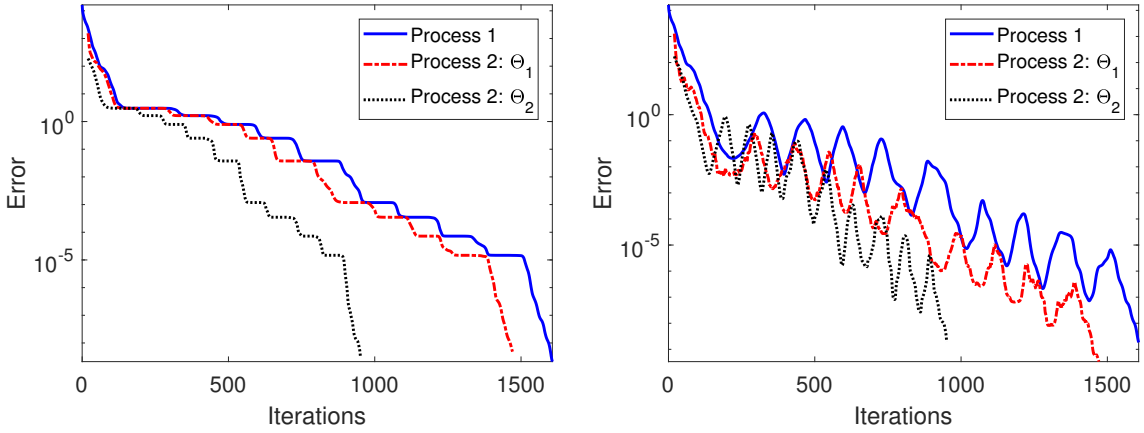
$$c^{(4)}(x,y) = \begin{cases} c_0 := 9 \times 10^5 & \text{if} \quad \lfloor (9x) \rfloor \in [1,7] \text{ and } \lfloor (9y) \rfloor \in [6,9] \\ 1 & \text{otherwise} \end{cases}$$

This time, with a maximum edge size $H_{\max} = 0.01$, we obtain a uniform mesh of 32 544 elements. The size of the matrix $\mathbf{A}$ is $16\,057 \times 16\,057$. The Galerkin solution and the initial algebraic error distribution over the domain Ω are depicted in Figure 2.16, while the global energy norm and the L-norm of the error during iterations are plotted in Figure 2.17. We start with a value $\Theta_1 = 0.81$



(a) Galerkin solution $u_h^{(3)}$ (b) Algebraic errors after 20 iterations

Figure 2.13 – *Galerkin solution and initial algebraic error distribution for test case n°3*



(a) Energy norm of the algebraic error (b) L-norm of the algebraic error

Figure 2.14 – *Error evolution for test case n°3*

that ensures that the size of the submatrix $\mathbf{A}_L$ is about one tenth of the size of the global matrix $\mathbf{A}$. While the blue and red curves of Figure 2.17 are very close to each other for the global energy norm, we observe that the red one is always below the blue one when it comes to the L-norm.
For this test case, we see that the high algebraic errors are concentrated on a rectangle. With the Dörfler rate Θ_1, we capture only an upper band of that rectangle. As a consequence, the curves associated to process 2 will not show any substantial improvement over those of process 1 with that setting.
On the other hand, increasing the value of the Dörfler rate and considering $\Theta_2 = 0.99$ ensures that the L-subdomain covers almost all the rectangle with large algebraic errors. We obtain a submatrix $\mathbf{A}_L$ whose size is about 20% of the size of the global matrix $\mathbf{A}$. Now, we see in Figure 2.17 that the curves of process 1 (blue) and process 2 (black) are distinct.

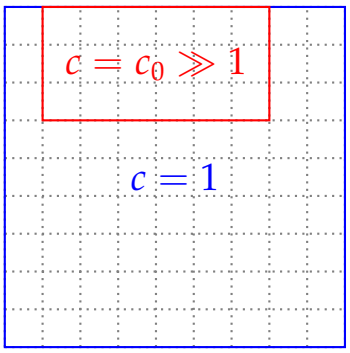
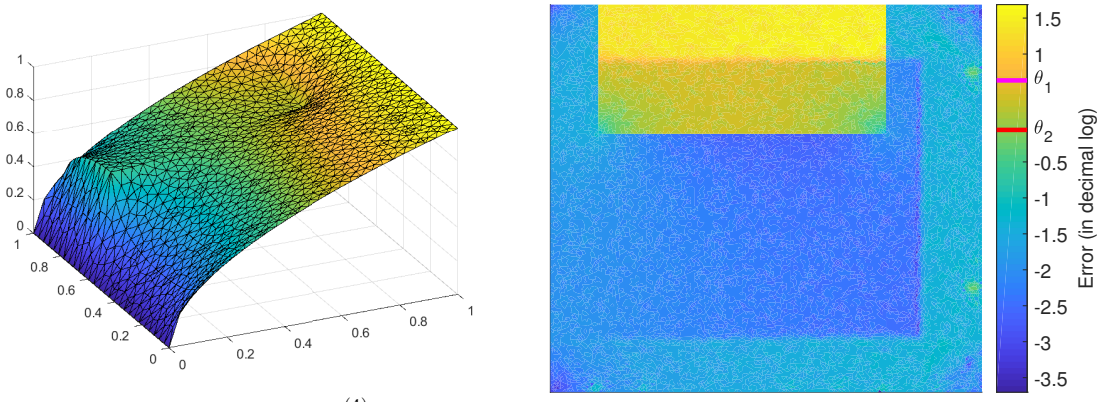
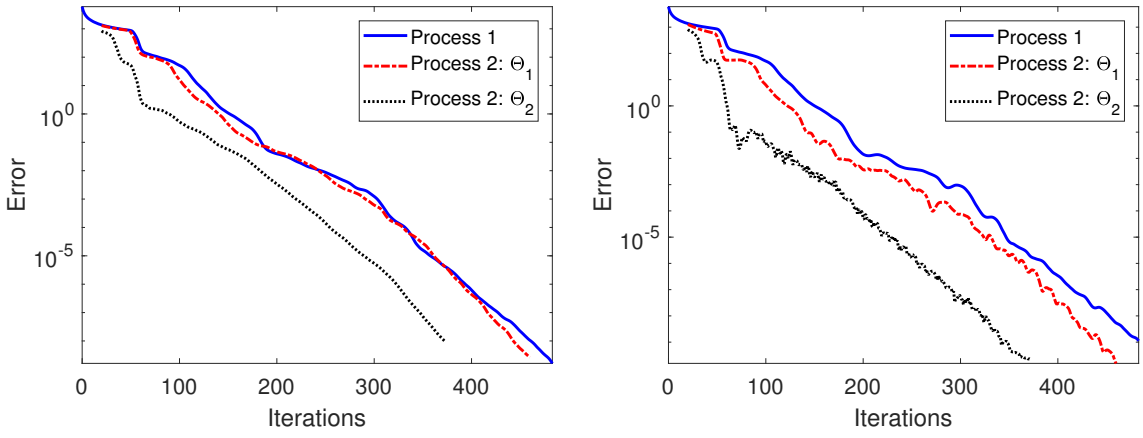


Figure 2.15 – *Configuration of the inhomogeneous diffusivity in test case n°4*



(a) Galerkin solution $u_h^{(4)}$ (b) Algebraic errors after 20 iterations

Figure 2.16 – *Galerkin solution and initial algebraic error distribution for test case n°4*



(a) Energy norm of the algebraic error (b) L-norm of the algebraic error

Figure 2.17 – *Error evolution for test case n°4*

Table 2.2 – *Test configuration and number of iterations for standard and adaptive processes with different values of the Dörfler rate Θ applied to diffusion problems.*

Configuration			Iterations	
Test case	Θ (as %)	n_L in % of n	it_{st}	it_{ada}
3	95	23.38	1587	1451
	99.93	69.98	1587	931
4	81	10	463	440
	99.99	21.68	463	353

In almost all of the test cases presented above (except test case n°3), we have voluntarily restricted ourselves to taking subdomains Ω_1 whose size n_L does not exceed 25% of n the size of the global matrix in order to avoid configurations where process 2 is too costly with respect to process 1. However, it is important to bear in mind that the ideal size of the L-subdomain cannot be always limited under a certain threshold and is linked to the distribution of the algebraic errors for the problem considered.

In the light of the above, we can draw the following conclusions:

- The solve process proposed in this paper is adaptive as it adjusts to the considered problem thanks to the information stemming from the a posteriori estimation of the algebraic error. It seems to perform well provided that Ω_1 is appropriately built, i.e. we mark and gather in Ω_1 enough (or all ideally) elements that really reflect the regions of the domain where the solution of the system is more delicate and requires some special local treatment.
- The adaptive process is better suited for test cases where there is a considerable discrepancy in errors and a concentration in space that is conducive to adaptivity (like (2.39) and (2.40)). In such cases, the elements' marking is facilitated by the fact that the high algebraic errors seem to properly represent the potential regions that are of interest to us in order to build Ω_1. Herein, one can see an analogy with adaptive refinement in Finite Element Methods.
- In addition, this procedure seems to perform poorly when the algebraic errors are widely spread in the domain. In such cases, it becomes costly to include all elements with significant errors inside Ω_1 since we will have to factorize the $\mathbf{A}_L$ submatrix. And when we fill that subdomain with only a part of those elements, the procedure yields unsatisfactory results as we can see with test case n°4. When we include all the region of large errors inside Ω_1, the adaptive procedure performs well as we observed in test cases n°1, n°2 and n°4.
- Another issue relates to the spread of the error through the whole domain. We notice that the adaptive procedure performs better when the area with large errors is contiguous. We can clearly see that by comparing test cases n°3 and n°4.

2.7 Conclusions

In this chapter, we have presented a new adaptive preconditioner for iterative solution of sparse linear systems arising from PDE problems that is used in combination with a specific initial guess and based on the estimated local distribution of the algebraic error. The proposed adaptive procedure aims at efficiently reducing the algebraic error norm by targeting the regions

where the algebraic error is high. As shown in numerical experiments, in the case of an important discrepancy in the algebraic error, when it is dominating in certain parts of the domain, notable speedups can be achieved with the proposed procedure: with a proper treatment on high-error regions, the number of iterations can be significantly diminished.

We are aware that there is a lot of more work to be done in order to derive a robust practically applicable and efficient procedure. Nevertheless, the present study has confirmed that the concept of adaptivity based on the (local) distribution of the algebraic error is worth considering. A follow-up direction could be to investigate a more general algorithm not necessarily requiring that the L-subdomain solve is carried out exactly (as suggested e.g. in [87]).

3

Adaptive approach of local preconditioning based on low rank approximation and a posteriori error estimates

Contents

Abstract

This chapter focuses on a class of application specific preconditioners for iteratively solving linear systems stemming from the discretization of partial differential equations (PDE). Adaptive error-based preconditioners have been previously suggested to reduce high local algebraic error. This type of preconditioners requires an exact factorization/inversion of a submatrix of the global main matrix. This operation can represent a serious computational impediment during the solve process. Therefore, we propose here alternatives, still in the same adaptive framework, that replace exact inverses by approximate ones. This could be a preferred option to consider when the areas with high algebraic errors are large and scattered.
For this purpose, one can employ robust preconditioning techniques, which bound the condition number from above e.g. LORASC, to avoid exactly inverting some large blocks of the matrix. In addition, we present the upper bounds for the condition number of the preconditioned operator with the resulting preconditioner.
Subsequently, we explain the concept, discuss the preconditioning costs incurred by both exact and approximate adaptive preconditioners. Finally, we assess the feasibility and the reliability of this alternative by some numerical experiments.
Keywords— Approximate inverse, inexact factorization, LORASC, condition number bounds, adaptivity, preconditioning

3.1 Introduction

In the previous chapter, we saw that the size of the L-subset can be large when the algebraic error is widely spread over the domain. Dealing with non-localized errors raises some issues. As the L-subset gets larger, the adaptive solve procedure becomes more delicate to use due to the direct solve to carry out on a sizeable matrix $\mathbf{A}_L$. Hence we are interested in a variant of the adaptive procedure where direct solves are replaced by approximate methods.

All direct methods for solving a linear system are built on the idea of reducing a problem to the solution of triangular systems by the standard back and forward substitution. These include methods based on Gaussian Elimination, QR decomposition, LU decomposition, Cholesky decomposition etc [77] Rounding error analysis has been extensively studied over the years for this kind of methods, e.g. [62, 77]. In fact, direct methods offer the advantage of robustness and are generally backward stable. Therefore, they are preferred over other classes of methods when a high level of accuracy is required.

That being said, one of the well known drawbacks of exact factorization methods is their computational cost. The associated complexity is too prohibitive. The other possible issue relates to the density of the factors. The sparsity of the matrix doesn't imply the sparsity of its factors. A classical example is when the matrix is sparse but its graph is connected, then the inversion/factorization generates fill-ins and the inverse/factors become dense.

On the other hand, there exists different ways to perform inexact or approximate inversion such as incomplete factorizations and low rank approximation. Admittedly, those techniques are always inferior to direct methods in terms of accuracy, but the computational effort is reduced, which make them more affordable in practice.

In this chapter, we take into account some of the approximation methods mentioned earlier in the state-of-the-art to propose an extension of the work initiated in Chapter 2 and derive similar

approximate adaptive procedures where exact solves are replaced by inexact ones. This process could be more convenient to carry out in practice for very large matrices as they allow to overcome the limits imposed by the exact factorization.

This chapter is organized as follows. We start by recalling the framework for error-based partitioned matrix preconditioning in Section 3.2. In Section 3.3, we give an example of an inverse approximation method called LORASC and describe how to incorporate it in the adaptive preconditioner. We also present the condition number's bounds for the preconditioned operator with both exact and approximate adaptive preconditioners in Section 3.4 and discuss the preconditioning costs in Section 3.5. In Section 3.7, some numerical results are provided and the conclusion remark is given in Section 3.8.

3.2 Preliminaries

Let us consider a self-adjoint PDE problem discretized by finite element or finite volume method on a polytopal domain. The equivalent system of linear algebraic equations is written

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b},$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a matrix of size $n \times n$, the right hand side $\mathbf{b}$ and the solution $\mathbf{x}$ are vectors of length n which is the number of degrees of freedom. The error analysis of the PDE allows to compute a posteriori error estimates for assessing the pertinence of the approximate solution at an early stage of the iterative solve process (iteration i). Some error estimates can even be decomposed in various components that identify different sources of errors [55, 46]. Among these, we focus on the algebraic error component. Given the values of this component on each mesh element, we would distinguish two sets of nodes:

- L: The set of nodes with significant a posteriori algebraic error estimates. Its size is denoted n_L.
- R: The set of nodes with negligible, or even nil a posteriori algebraic error estimates. Its size is denoted n_R.

Note that R is complementary of the set L in $[\![1, n]\!]$. On the basis of such a classification, one can split the linear system to obtain a 2×2 block partitioning:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_L & \mathbf{A}_{LR} \\ \mathbf{A}_{RL} & \mathbf{A}_R \end{pmatrix} \tag{3.1}$$

More details about the sorting and splitting processes can be found respectively in Section 2.4.2 and Section 2.6.1 of Chapter 2.

As far as the algebraic error is concerned, the starting assumption is as expressed in Corollary 2.1 of Chapter 2:

$$(\mathbf{x} - \mathbf{x}^{(i)})_L^{\mathrm{T}} \cdot \mathbf{A}_L \cdot (\mathbf{x} - \mathbf{x}^{(i)})_L \geq (\mathbf{x} - \mathbf{x}^{(i)})^{\mathrm{T}} \cdot \mathbf{A}_p^{(1)} \cdot (\mathbf{x} - \mathbf{x}^{(i)}) \gg (\mathbf{x} - \mathbf{x}^{(i)})^{\mathrm{T}} \cdot \mathbf{A}_p^{(2)} \cdot (\mathbf{x} - \mathbf{x}^{(i)})$$

where $\mathbf{A}_p^{(1)}$ and $\mathbf{A}_p^{(2)}$ are the extensions of the local stiffness matrices, $\mathbf{A}^{(1)}$ discretized on subdomain Ω_1 and $\mathbf{A}^{(2)}$ discretized on subdomain Ω_2, to the whole domain Ω. As explained in

Section 2.5 of Chapter 2, an initial guess $\mathbf{x}^{(0)}$ and a preconditioner $\mathbf{M}$ that may ensure the decay of high local algebraic errors during a PCG solve have the general shape:

$$\mathbf{x}^{(0)} = \begin{bmatrix} \mathbf{A}_L^{-1} \cdot (\mathbf{b}_L - \mathbf{A}_{LR} \cdot \mathbf{x}_R^{(0)}) \\ \mathbf{x}_R^{(0)} \end{bmatrix} , \quad \mathbf{M} := \begin{pmatrix} \mathbf{M}_1 & \mathbf{M}_1 \mathbf{A}_L^{-\mathrm{T}} \mathbf{A}_{LR} \\ \mathbf{A}_{RL} \mathbf{A}_L^{-1} \mathbf{M}_1 & \mathbf{M}_2 + \mathbf{A}_{RL} \mathbf{A}_L^{-1} \mathbf{A}_{LR} \end{pmatrix} ; \tag{3.2}$$

where $\mathbf{x}_R^{(0)} \in \mathbb{R}^{n_R}$ is an arbitrary vector, and $\mathbf{M}_1$, $\mathbf{M}_2$ are two SPD preconditioners of sizes n_L and n_R respectively. Then a particular case for the preconditioner, for which we can establish an equivalence with a classical Schur complement procedure (see Theorem 2.1 in Chapter 2), is when $\mathbf{M}_1 = \mathbf{A}_L$:

$$\mathbf{M} := \begin{pmatrix} \mathbf{A}_L & \mathbf{A}_{LR} \\ \mathbf{A}_{RL} & \mathbf{M}_2 + \mathbf{A}_{RL} \mathbf{A}_L^{-1} \mathbf{A}_{LR} \end{pmatrix} . \tag{3.3}$$

Since we are using PCG, we need to provide to the solver either the inverse of the preconditioner $\mathbf{M}^{-1}$ or its application to a vector. As one can notice in (3.2) or (3.3) this requires inverting $\mathbf{A}_L$ or solving systems involving this matrix.

In the previous chapter, it was assumed that an exact Cholesky factorization of $\mathbf{A}_L$ is known, therefore solving such systems can be achieved by forward and backward substitutions. However this assumption may not hold for all test cases, including those where the algebraic error is widely spread over the domain, because the matrix $\mathbf{A}_L$ becomes too large to be factored at an affordable cost. In Section 3.3, we investigate means to circumvent this difficulty by approximating $\mathbf{A}_L^{-1}$. Doing so introduces approximation errors that prevent the residual from remaining nil on the L part of the matrix (according to partitioning of (3.1)). Therefore, we shall rather focus on the quality of the approximate adaptive preconditioner with $\mathbf{A}_L^{-1}$ and whether or not we are able to prove some bounds for the condition number of the global preconditioned matrix $\mathbf{M}^{-1}\mathbf{A}$.
In what follows, we give an example of a preconditioner referred to as LORASC that we could incorporate in the adaptive preconditioner. Some of the advantages of LORASC method are the following: it allows to bound the condition number of the preconditioned operator, this preconditioner is fully algebraic and requires no information other than the matrix.

3.3 A low rank approximation on $\mathbf{A}_L$

In this section, we detail the building process of this approximation and its injection in the adaptive preconditioner. This low rank approximation technique has already been suggested in [67] for constructing preconditioners for Schur complement matrices and is based on Wielandt's deflation (see [122] and references therein). The procedure described on [67] requires permuting the matrix in order to obtain a block-arrow structure and focuses on the last diagonal block that corresponds to the separator between the disjoint domains. Much of the computational work will afterwards be performed on this particular block.

First, we describe how to build the approximate inverse of the matrix $\mathbf{A}_L$ with LORASC preconditioner [67]. In addition to the ordering based on the algebraic error distribution that gives a 2×2 partitioning of (3.1) where L, (resp. R) stand for the degrees of freedom where large (resp. small) errors were observed, we consider a reordering (k-way algorithm for example) on $\mathbf{A}_L$ that

yields a block arrow form,

$$\mathbf{A}_L = \begin{pmatrix} \mathbf{A}_{11}^L & & & \mathbf{A}_{1\Gamma}^L \\ & \ddots & & \vdots \\ & & \mathbf{A}_{NN}^L & \mathbf{A}_{N\Gamma}^L \\ \mathbf{A}_{\Gamma 1}^L & \dots & \mathbf{A}_{\Gamma N}^L & \mathbf{A}_{\Gamma\Gamma}^L \end{pmatrix}. \tag{3.4}$$

Then a factorization of $\mathbf{A}_L$ is,

$$\mathbf{A}_L = \begin{pmatrix} \mathbf{A}_{11}^L & & & \\ & \ddots & & \\ & & \mathbf{A}_{NN}^L & \\ \mathbf{A}_{\Gamma 1}^L & \dots & \mathbf{A}_{\Gamma N}^L & \mathbf{S}^L \end{pmatrix} \begin{pmatrix} (\mathbf{A}_{11}^L)^{-1} & & & \\ & \ddots & & \\ & & (\mathbf{A}_{NN}^L)^{-1} & \\ & & & (\mathbf{S}^L)^{-1} \end{pmatrix} \begin{pmatrix} \mathbf{A}_{11}^L & & & \mathbf{A}_{1\Gamma}^L \\ & \ddots & & \vdots \\ & & \mathbf{A}_{NN}^L & \mathbf{A}_{N\Gamma}^L \\ & & & \mathbf{S}^L \end{pmatrix}, \tag{3.5}$$

with $\mathbf{S}^L := \mathbf{A}_{\Gamma\Gamma}^L - \sum_{j=1}^{N} \mathbf{A}_{\Gamma j}^L (\mathbf{A}_{jj}^L)^{-1} \mathbf{A}_{j\Gamma}^L$. Likewise, we can express the inverse of $\mathbf{A}_L$ as,

$$\mathbf{A}_L^{-1} = \begin{pmatrix} I & & & -(\mathbf{A}_{11}^L)^{-1}\mathbf{A}_{1\Gamma}^L \\ & \ddots & & \vdots \\ & & I & -(\mathbf{A}_{NN}^L)^{-1}\mathbf{A}_{N\Gamma}^L \\ & & & I \end{pmatrix} \begin{pmatrix} (\mathbf{A}_{11}^L)^{-1} & & & \\ & \ddots & & \\ & & (\mathbf{A}_{NN}^L)^{-1} & \\ & & & (\mathbf{S}^L)^{-1} \end{pmatrix} \begin{pmatrix} I & & & \\ & \ddots & & \\ & & I & \\ -\mathbf{A}_{\Gamma 1}^L(\mathbf{A}_{11}^L)^{-1} & \dots & -\mathbf{A}_{\Gamma N}^L(\mathbf{A}_{NN}^L)^{-1} & I \end{pmatrix}. \tag{3.6}$$

For a sufficiently large N, we may assume that the $(\mathbf{A}_{jj}^L)^{-1}$ are computable for $1 \leq j \leq N$, and thus $\mathbf{S}^L$ as well. That being said, as N increases, $\mathbf{S}^L$ gets denser and it becomes more tedious to compute $(\mathbf{S}^L)^{-1}$. For that reason, the authors of [67] decided to derive an approximation $\widetilde{\mathbf{A}}_L^{-1}$ of $\mathbf{A}_L^{-1}$ by replacing $(\mathbf{S}^L)^{-1}$ in the formula above by $(\widetilde{\mathbf{S}}^L)^{-1}$ a corrected low rank approximation of $(\mathbf{S}^L)^{-1}$.

$$\widetilde{\mathbf{A}}_L^{-1} = \begin{pmatrix} I & & & -(\mathbf{A}_{11}^L)^{-1}\mathbf{A}_{1\Gamma}^L \\ & \ddots & & \vdots \\ & & I & -(\mathbf{A}_{NN}^L)^{-1}\mathbf{A}_{N\Gamma}^L \\ & & & I \end{pmatrix} \begin{pmatrix} (\mathbf{A}_{11}^L)^{-1} & & & \\ & \ddots & & \\ & & (\mathbf{A}_{NN}^L)^{-1} & \\ & & & (\widetilde{\mathbf{S}}^L)^{-1} \end{pmatrix} \begin{pmatrix} I & & & \\ & \ddots & & \\ & & I & \\ -\mathbf{A}_{\Gamma 1}^L(\mathbf{A}_{11}^L)^{-1} & \dots & -\mathbf{A}_{\Gamma N}^L(\mathbf{A}_{NN}^L)^{-1} & \mathbf{I} \end{pmatrix} \tag{3.7}$$

The first component for approximating $(\mathbf{S}^L)^{-1}$ is $(\mathbf{A}_{\Gamma\Gamma}^L)^{-1}$. This suffices to approximate the maximum eigenvalues of $\mathbf{S}^L$ as the following upper bound holds.

Lemma 3.1. *(Theorem 3.1 in [67]) Let $\widetilde{\mathbf{A}}_L^{-1}$ as defined in (3.7) with $\widetilde{\mathbf{S}}^L = \mathbf{A}_{\Gamma\Gamma}^L$. Then for every eigenvalue λ of $\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_L$, we have:*

$$\lambda \leq 1$$

Proof. See [67, Theorem 3.1]. □

Obviously, a good approximation $\widetilde{\mathbf{A}}_L$ of $\mathbf{A}_L$ is such that the matrix $\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_L$ is close to identity.

To this end, one could consider ensuring that the eigenvalues of the first are not so far apart from those of the second. In this respect, the result of Lemma 3.1 can be seen as a first stage of the

approximation process, it prevents the eigenvalues of $\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_L$ from exceeding 1.

The second stage consists in bounding those eigenvalues from below. One can impose a minimum threshold ϵ for the smallest eigenvalue of $\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_L$. This amounts to imposing the same threshold for the eigenvalues of $(\widetilde{\mathbf{S}}^L)^{-1}\mathbf{S}^L$. For that, the next component of the approximation is constructed from the eigenvalues and eigenvectors of $(\mathbf{A}_{\Gamma\Gamma}^L)^{-1}\mathbf{S}^L$ and will serve to approximate the smallest eigenvalues and deflate them.

Let $\lambda_1, \ldots, \lambda_p$ be the eigenvalues of $(\mathbf{A}_{\Gamma\Gamma}^L)^{-1}\mathbf{S}^L$ sorted in an increasing order and $\mathbf{v}_1, \ldots, \mathbf{v}_p$ be the associated $\mathbf{A}_{\Gamma\Gamma}^L$-orthonormal eigenvectors. Let also $i \in [\![1, p]\!]$ such that $\lambda_j < \epsilon \ , \quad \forall j \leq i$. We define $\mathbf{E}_i := [\mathbf{v}_1 \mathbf{v}_2 \ldots \mathbf{v}_i]$ and $\mathbf{\Sigma}_i = [\sigma_1 \sigma_2 \ldots \sigma_i]$ with $\sigma_j = \dfrac{\epsilon - \lambda_j}{\lambda_j} \ , \quad \forall j \leq i \ .$

Lemma 3.2. *(Theorem 3.1 in [67]) For* $(\widetilde{\mathbf{S}}^L)^{-1} = (\mathbf{A}_{\Gamma\Gamma}^L)^{-1} + \mathbf{E}_i\mathbf{\Sigma}_i\mathbf{E}_i^T$*, the spectrum of* $(\widetilde{\mathbf{S}}^L)^{-1}\mathbf{S}^L$ *is composed of* $\lambda_1, \ldots, \lambda_i, 1$ *and is bounded by* ϵ *and* 1*:*

$$\mathrm{Sp}((\widetilde{\mathbf{S}}^L)^{-1}\mathbf{S}^L) = \{\epsilon, \lambda_{i+1}, \ldots, \lambda_p\} \subset [\epsilon, 1] \tag{3.8}$$

Proof. See [67, Theorem 3.1]. □

Hence, Lemma 3.2 allows to show that with the approximation $(\widetilde{\mathbf{S}}^L)^{-1} = (\mathbf{A}_{\Gamma\Gamma}^L)^{-1} + \mathbf{E}_i\mathbf{\Sigma}_i\mathbf{E}_i^T$ the corresponding $\widetilde{\mathbf{A}}_L$ is such that the eigenvalues of $\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_L$ are between ϵ and 1 [67]. As a consequence, the choice of ϵ has a strong influence on the quality of the approximation. Indeed, the closest ϵ is to 1, the more accurate the approximate inverse $\widetilde{\mathbf{A}}_L^{-1}$ will be.

Now if we replace $\mathbf{A}_L^{-1}$ by $\widetilde{\mathbf{A}}_L^{-1}$ in (3.2), we obtain the general shape of the approximate preconditioner $\widetilde{\mathbf{M}}_{\mathrm{G}}$:

$$\widetilde{\mathbf{M}}_{\mathrm{G}}(\widetilde{\mathbf{A}}_L, \mathbf{M}_1, \mathbf{M}_2) := \begin{pmatrix} \mathbf{M}_1 & \mathbf{M}_1\widetilde{\mathbf{A}}_L^{-T}\mathbf{A}_{LR} \\ \mathbf{A}_{RL}\widetilde{\mathbf{A}}_L^{-1}\mathbf{M}_1 & \mathbf{M}_2 + \mathbf{A}_{RL}\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_{LR} \end{pmatrix} . \tag{3.9}$$

Similarly for (3.3), we obtain the approximate preconditioner in the particular case $\widetilde{\mathbf{M}}_{\mathrm{P}}$:

$$\widetilde{\mathbf{M}}_{\mathrm{P}}(\widetilde{\mathbf{A}}_L, \mathbf{M}_2) := \begin{pmatrix} \widetilde{\mathbf{A}}_L & \mathbf{A}_{LR} \\ \mathbf{A}_{RL} & \mathbf{M}_2 + \mathbf{A}_{RL}\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_{LR} \end{pmatrix} . \tag{3.10}$$

We can notice in both formulae above that we have taken out the computational difficulty relating to the inverse of $\mathbf{A}_L$, as building or applying $\widetilde{\mathbf{M}}^{-1}$ now depends only on the computation of $\mathbf{M}_1^{-1}$ and $\mathbf{M}_2^{-1}$ for (3.9) or $\mathbf{M}_2^{-1}$ for (3.10) as well as $\widetilde{\mathbf{A}}_L^{-1}$.

In the sequel, we show the condition number bounds for the preconditioned operator when using the approximate variant of (3.10) and compare it to the bound obtained with the exact adaptive preconditioner of (3.3).

3.4 Some condition number bounds for the exact and the approximate adaptive preconditioners

In this section, we show that $\kappa(\mathbf{M}^{-1}\mathbf{A})$ the condition number of the preconditioned matrix by the adaptive preconditioner is bounded from above. For that, we bound $g(\kappa(\mathbf{M}^{-1}\mathbf{A}))$ where g is the function defined by

$$\begin{aligned} g\colon [1,\infty[&\to [2,\infty[\\ x &\mapsto x + \frac{1}{x}. \end{aligned}$$

As g is a monotonically increasing function, bounding $g(\kappa(\mathbf{M}^{-1}\mathbf{A}))$ implies that $\kappa(\mathbf{M}^{-1}\mathbf{A})$ is bounded as well.

We start with the exact adaptive preconditioner defined in (3.3).

Lemma 3.3. *Let* $\mathbf{W} := \begin{pmatrix} \mathbf{W}_L & \mathbf{0} \\ \mathbf{A}_{RL}\mathbf{W}_L^{-\mathrm{T}} & \mathbf{W}_2 \end{pmatrix}$ *be the split form of the exact adaptive preconditioner, i.e.*

$$\mathbf{M} = \mathbf{W}\mathbf{W}^{\mathrm{T}} = \mathbf{M} = \begin{pmatrix} \mathbf{A}_L & \mathbf{A}_{LR} \\ \mathbf{A}_{RL} & \mathbf{M}_2 + \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR} \end{pmatrix};$$

with $\mathbf{W}_L\mathbf{W}_L^{\mathrm{T}} = \mathbf{A}_L$ *and* $\mathbf{W}_2\mathbf{W}_2^{\mathrm{T}} = \mathbf{M}_2$, $\mathbf{M}_2$ *is an arbitrary preconditioner for the Schur complement* $\mathbf{S} := \mathbf{A}_R - \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR}$. *Then we have:*

$$\kappa(\mathbf{W}^{-1}\mathbf{A}\mathbf{W}^{-\mathrm{T}}) + (\kappa(\mathbf{W}^{-1}\mathbf{A}\mathbf{W}^{-\mathrm{T}}))^{-1} + 2 \leq (1 + \lambda_{\max}(\mathbf{W}_2^{-1}\mathbf{S}\mathbf{W}_2^{\mathrm{T}}))(1 + \lambda_{\min}^{-1}(\mathbf{W}_2^{-1}\mathbf{S}\mathbf{W}_2^{-\mathrm{T}})); \quad (3.11)$$

and:

$$\kappa(\mathbf{W}^{-1}\mathbf{A}\mathbf{W}^{-\mathrm{T}}) + (\kappa(\mathbf{W}^{-1}\mathbf{A}\mathbf{W}^{-\mathrm{T}}))^{-1} + 2 \leq (1 + \lambda_{\max}(\mathbf{W}_2^{-1}\mathbf{A}_R\mathbf{W}_2^{\mathrm{T}}))(1 + \lambda_{\min}^{-1}(\mathbf{W}_2^{-1}\mathbf{S}\mathbf{W}_2^{-\mathrm{T}})). \quad (3.12)$$

Proof. As $\mathbf{W}^{-1} = \begin{pmatrix} \mathbf{W}_L^{-1} & \mathbf{0} \\ -\mathbf{W}_2^{-1}\mathbf{A}_{RL}\mathbf{A}_L^{-1} & \mathbf{W}_2^{-1} \end{pmatrix}$, we obtain $\mathbf{W}^{-1}\mathbf{A}\mathbf{W}^{-\mathrm{T}} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_2^{-1}\mathbf{S}\mathbf{W}_2^{-\mathrm{T}} \end{pmatrix}$.
$\mathbf{W}^{-1}\mathbf{A}\mathbf{W}^{-\mathrm{T}}$ is a symmetric matrix and by consequence we can apply Lemma 8.6 of [22]:

$$\begin{aligned} \lambda_{\max}(\mathbf{W}^{-1}\mathbf{A}\mathbf{W}^{-\mathrm{T}}) + \lambda_{\min}(\mathbf{W}^{-1}\mathbf{A}\mathbf{W}^{-\mathrm{T}}) &\leq \lambda_{\max}(\mathbf{W}_L^{-1}\mathbf{A}_L\mathbf{W}_L^{-\mathrm{T}}) + \lambda_{\max}(\mathbf{W}_2^{-1}\mathbf{S}\mathbf{W}_2^{-\mathrm{T}}) \\ &\leq 1 + \lambda_{\max}(\mathbf{W}_2^{-1}\mathbf{S}\mathbf{W}_2^{-\mathrm{T}}). \end{aligned} \quad (3.13)$$

Since $\mathbf{A}_R - \mathbf{S} = \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR}$ is an SPSD matrix, it also holds that

$$\lambda_{\max}(\mathbf{W}_2^{-1}\mathbf{S}\mathbf{W}_2^{-\mathrm{T}}) \leq \lambda_{\max}(\mathbf{W}_2^{-1}\mathbf{A}_R\mathbf{W}_2^{-\mathrm{T}}).$$

Consequently,

$$\lambda_{\max}(\mathbf{W}^{-1}\mathbf{A}\mathbf{W}^{-\mathrm{T}}) + \lambda_{\min}(\mathbf{W}^{-1}\mathbf{A}\mathbf{W}^{-\mathrm{T}}) \leq 1 + \lambda_{\max}(\mathbf{W}_2^{-1}\mathbf{A}_R\mathbf{W}_2^{-\mathrm{T}}). \quad (3.14)$$

Similarly, for the matrix $\mathbf{W}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{W} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_2^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{W}_2 \end{pmatrix}$, which is symmetric, the same lemma mentioned above yields:

$$\begin{aligned} \lambda_{\max}(\mathbf{W}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{W}) + \lambda_{\min}(\mathbf{W}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{W}) &\leq \lambda_{\max}(\mathbf{W}_L^{\mathrm{T}}\mathbf{A}_L^{-1}\mathbf{W}_L) + \lambda_{\max}(\mathbf{W}_2^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{W}_2) \\ &\leq 1 + \lambda_{\min}^{-1}(\mathbf{W}_2^{-1}\mathbf{S}\mathbf{W}_2^{-\mathrm{T}}). \end{aligned} \tag{3.15}$$

By multiplying Inequalities (3.13) and (3.15), we obtain:

$$\kappa(\mathbf{W}^{-1}\mathbf{A}\mathbf{W}^{-\mathrm{T}}) + (\kappa(\mathbf{W}^{-1}\mathbf{A}\mathbf{W}^{-\mathrm{T}}))^{-1} + 2 \leq (1 + \lambda_{\max}(\mathbf{W}_2^{-1}\mathbf{S}\mathbf{W}_2^{\mathrm{T}}))(1 + \lambda_{\min}^{-1}(\mathbf{W}_2^{-1}\mathbf{S}\mathbf{W}_2^{-\mathrm{T}})).$$

On the other hand, by multiplying Inequalities (3.14) and (3.15), we obtain:

$$\kappa(\mathbf{W}^{-1}\mathbf{A}\mathbf{W}^{-\mathrm{T}}) + (\kappa(\mathbf{W}^{-1}\mathbf{A}\mathbf{W}^{-\mathrm{T}}))^{-1} + 2 \leq (1 + \lambda_{\max}(\mathbf{W}_2^{-1}\mathbf{A}_R\mathbf{W}_2^{\mathrm{T}}))(1 + \lambda_{\min}^{-1}(\mathbf{W}_2^{-1}\mathbf{S}\mathbf{W}_2^{-\mathrm{T}})).$$

□

Considering a preconditioner that bounds the eigenvalues of $\mathbf{S}$, for example a low-rank approximation for $\mathbf{S}$: $\mathbf{M}_2^{-1} = \mathbf{W}_2^{-\mathrm{T}}\mathbf{W}_2^{-1} = \widetilde{\mathbf{S}}^{-1}$ such that for a chosen $\epsilon > 0$:

$$\lambda_{\max}(\mathbf{W}_2^{-1}\mathbf{S}\mathbf{W}_2^{-\mathrm{T}}) \leq 1; \quad \text{and} \quad \lambda_{\min}(\mathbf{W}_2^{-1}\mathbf{S}\mathbf{W}_2^{-\mathrm{T}}) \geq \epsilon;$$

in this case, Inequality (3.12) yields

$$\kappa(\mathbf{W}^{-1}\mathbf{A}\mathbf{W}^{-\mathrm{T}}) + (\kappa(\mathbf{W}^{-1}\mathbf{A}\mathbf{W}^{-\mathrm{T}}))^{-1} \leq \frac{2}{\epsilon}. \tag{3.16}$$

Now, let us demonstrate analogous bounds for the approximate adaptive preconditioner defined in (3.10).

Lemma 3.4. *Let* $\widetilde{\mathbf{W}} := \begin{pmatrix} \widetilde{\mathbf{W}}_L & \mathbf{0} \\ \mathbf{A}_{RL}\widetilde{\mathbf{W}}_L^{-\mathrm{T}} & \mathbf{W}_2 \end{pmatrix}$ *be the split form of the approximate adaptive preconditioner, i.e.* $\mathbf{M} = \widetilde{\mathbf{W}}\widetilde{\mathbf{W}}^{\mathrm{T}}$*; with* $\widetilde{\mathbf{W}}_L\widetilde{\mathbf{W}}_L^{\mathrm{T}} = \widetilde{\mathbf{A}}_L$ *and* $\mathbf{W}_2\mathbf{W}_2^{\mathrm{T}} = \mathbf{M}_2$, $\mathbf{M}_2$ *is an arbitrary preconditioner for the Schur complement* $\mathbf{S} := \mathbf{A}_R - \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR}$*. We denote*

$$\widehat{\mathbf{S}} := \mathbf{W}_2^{-1}\check{\mathbf{S}}\mathbf{W}_2^{-\mathrm{T}}$$

where

$$\check{\mathbf{S}} := \mathbf{A}_R - \mathbf{A}_{RL}(2\widetilde{\mathbf{A}}_L^{-1} - \widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{A}}_L^{-1})\mathbf{A}_{LR}$$

then

$$\begin{aligned} &\kappa(\widetilde{\mathbf{W}}^{-1}\mathbf{A}\widetilde{\mathbf{W}}^{-\mathrm{T}}) + (\kappa(\widetilde{\mathbf{W}}^{-1}\mathbf{A}\widetilde{\mathbf{W}}^{-\mathrm{T}}))^{-1} + 2 \leq \\ &\left(\lambda_{\max}(\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{W}}_L^{-\mathrm{T}}) + \lambda_{\max}(\widehat{\mathbf{S}})\right)\left(\lambda_{\min}^{-1}(\mathbf{W}_2^{-1}\mathbf{S}\mathbf{W}_2^{-\mathrm{T}}) + \lambda_{\min}^{-1}(\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{W}}_L^{-\mathrm{T}})\right. \\ &\left.+\lambda_{\max}(\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_{LR}\mathbf{S}^{-1}\mathbf{A}_{RL}\widetilde{\mathbf{W}}_L^{-\mathrm{T}}) \times (\lambda_{\max}(\widetilde{\mathbf{W}}_L^{\mathrm{T}}\mathbf{A}_L^{-1}\widetilde{\mathbf{W}}_L) - 1)^2\right) \end{aligned} \tag{3.17}$$

If in addition, the approximation on the L-block is such that

$$\lambda_{\max}(\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{W}}_L^{-\mathrm{T}}) \leq 2;$$

then the bound becomes

$$\begin{aligned}&\kappa(\widetilde{\mathbf{W}}^{-1}\mathbf{A}\widetilde{\mathbf{W}}^{-\mathrm{T}}) + (\kappa(\widetilde{\mathbf{W}}^{-1}\mathbf{A}\widetilde{\mathbf{W}}^{-\mathrm{T}}))^{-1} + 2 \leq \\ &\left(2 + \lambda_{\max}(\mathbf{W}_2^{-1}\mathbf{A}_R\mathbf{W}_2^{-\mathrm{T}})\right)\left(\lambda_{\min}^{-1}(\mathbf{W}_2^{-1}\mathbf{S}\mathbf{W}_2^{-\mathrm{T}}) + \lambda_{\min}^{-1}(\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{W}}_L^{-\mathrm{T}})\right. \\ &\left.+\lambda_{\max}(\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_{LR}\mathbf{S}^{-1}\mathbf{A}_{RL}\widetilde{\mathbf{W}}_L^{-\mathrm{T}}) \times (\lambda_{\max}(\widetilde{\mathbf{W}}_L^{\mathrm{T}}\mathbf{A}_L^{-1}\widetilde{\mathbf{W}}_L) - 1)^2\right)\end{aligned} \tag{3.18}$$

Proof. We have

$$\widetilde{\mathbf{W}}^{-1}\mathbf{A}\widetilde{\mathbf{W}}^{-\mathrm{T}} = \begin{pmatrix} \widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{W}}_L^{-\mathrm{T}} & -\widetilde{\mathbf{W}}_L^{-1}(\mathbf{A}_L\widetilde{\mathbf{A}}_L^{-1} - \mathbf{I})\mathbf{A}_{LR}\mathbf{W}_2^{-\mathrm{T}} \\ \mathbf{W}_2^{-1}\mathbf{A}_{RL}(\mathbf{I} - \widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_L)\widetilde{\mathbf{W}}_L^{-\mathrm{T}} & \widehat{\mathbf{S}} \end{pmatrix}.$$

$\widetilde{\mathbf{W}}^{-1}\mathbf{A}\widetilde{\mathbf{W}}^{-\mathrm{T}}$ is a symmetric, thus we can apply Lemma 8.6 of [22]:

$$\lambda_{\max}(\widetilde{\mathbf{W}}^{-1}\mathbf{A}\widetilde{\mathbf{W}}^{-\mathrm{T}}) + \lambda_{\min}(\widetilde{\mathbf{W}}^{-1}\mathbf{A}\widetilde{\mathbf{W}}^{-\mathrm{T}}) \leq \lambda_{\max}(\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{W}}_L^{-\mathrm{T}}) + \lambda_{\max}(\widehat{\mathbf{S}}) \tag{3.19}$$

On the other hand, we denote

$$\mathbf{Q} := \widetilde{\mathbf{W}}_L^{\mathrm{T}}\mathbf{A}_L^{-1}\widetilde{\mathbf{W}}_L + (\widetilde{\mathbf{W}}_L^{\mathrm{T}}\mathbf{A}_L^{-1}\widetilde{\mathbf{W}}_L - \mathbf{I})\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_{LR}\mathbf{S}^{-1}\mathbf{A}_{RL}\widetilde{\mathbf{W}}_L^{-\mathrm{T}}(\widetilde{\mathbf{W}}_L^{\mathrm{T}}\mathbf{A}_L\widetilde{\mathbf{W}}_L - \mathbf{I}),$$

then

$$\widetilde{\mathbf{W}}^{\mathrm{T}}\mathbf{A}^{-1}\widetilde{\mathbf{W}} = \begin{pmatrix} \mathbf{Q} & (\widetilde{\mathbf{W}}_L^{-1} - \widetilde{\mathbf{W}}_L^{\mathrm{T}}\mathbf{A}_L^{-1})\mathbf{A}_{LR}\mathbf{S}^{-1}\mathbf{W}_2 \\ \mathbf{W}_2^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{A}_{RL}(\widetilde{\mathbf{W}}_L^{-\mathrm{T}} - \mathbf{A}_L^{-1}\widetilde{\mathbf{W}}_L) & \mathbf{W}_2^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{W}_2 \end{pmatrix};$$

and consequently

$$\lambda_{\max}(\widetilde{\mathbf{W}}^{\mathrm{T}}\mathbf{A}^{-1}\widetilde{\mathbf{W}}) + \lambda_{\min}(\widetilde{\mathbf{W}}^{\mathrm{T}}\mathbf{A}^{-1}\widetilde{\mathbf{W}}) \leq \lambda_{\max}(\mathbf{Q}) + \lambda_{\max}(\mathbf{W}_2^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{W}_2)$$

From the definition of $\mathbf{Q}$, we can deduce that:

$$\lambda_{\max}(\mathbf{Q}) \leq \lambda_{\max}(\widetilde{\mathbf{W}}_L^{\mathrm{T}}\mathbf{A}_L^{-1}\widetilde{\mathbf{W}}_L) + \lambda_{\max}(\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_{LR}\mathbf{S}^{-1}\mathbf{A}_{RL}\widetilde{\mathbf{W}}_L^{-\mathrm{T}}) \times \lambda_{\max}^2(\widetilde{\mathbf{W}}_L^{\mathrm{T}}\mathbf{A}_L^{-1}\widetilde{\mathbf{W}}_L - \mathbf{I})$$

Combining those two latter inequalities yields

$$\begin{aligned}\lambda_{\min}^{-1}(\widetilde{\mathbf{W}}^{-1}\mathbf{A}\widetilde{\mathbf{W}}^{-\mathrm{T}}) + \lambda_{\max}^{-1}(\widetilde{\mathbf{W}}^{-1}\mathbf{A}\widetilde{\mathbf{W}}^{-\mathrm{T}}) \leq &\lambda_{\min}^{-1}(\mathbf{W}_2^{-1}\mathbf{S}\mathbf{W}_2^{-\mathrm{T}}) + \lambda_{\min}^{-1}(\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{W}}_L^{-\mathrm{T}}) + \\ &\lambda_{\max}(\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_{LR}\mathbf{S}^{-1}\mathbf{A}_{RL}\widetilde{\mathbf{W}}_L^{-\mathrm{T}}) \times (\lambda_{\max}(\widetilde{\mathbf{W}}_L^{\mathrm{T}}\mathbf{A}_L^{-1}\widetilde{\mathbf{W}}_L) - 1)^2\end{aligned} \tag{3.20}$$

By multiplying (3.19) by (3.20), we obtain Inequality (3.17).

Now, if $\lambda_{\max}(\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{W}}_L^{-\mathrm{T}}) \leq 2$, let $(\lambda, \mathbf{y})$ be an eigenpair of $\widehat{\mathbf{S}}$. We have:

$$\begin{aligned}\langle \widehat{\mathbf{S}}\cdot\mathbf{y},\mathbf{y}\rangle &= \langle \mathbf{W}_2^{-1}\mathbf{A}_R\mathbf{W}_2^{-\mathrm{T}}\cdot\mathbf{y},\mathbf{y}\rangle - \mathbf{y}^{\mathrm{T}}\cdot\mathbf{W}_2^{-1}\mathbf{A}_{RL}(2\widetilde{\mathbf{A}}_L^{-1}-\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{A}}_L^{-1})\mathbf{A}_{LR}\mathbf{W}_2^{-\mathrm{T}}\cdot\mathbf{y}\\ &= \langle \mathbf{W}_2^{-1}\mathbf{A}_R\mathbf{W}_2^{-\mathrm{T}}\cdot\mathbf{y},\mathbf{y}\rangle - (\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_{LR}\mathbf{W}_2^{-\mathrm{T}}\cdot\mathbf{y})^{\mathrm{T}}(2\mathbf{I}-\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{W}}_L^{-\mathrm{T}})(\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_{LR}\mathbf{W}_2^{-\mathrm{T}}\cdot\mathbf{y})\end{aligned} \tag{3.21}$$

Let $\mathbf{z}\in\mathbb{R}^{n_L}$, we have: $\langle(2\mathbf{I}-\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{W}}_L^{-\mathrm{T}})\cdot\mathbf{z},\mathbf{z}\rangle = \langle 2\mathbf{z},\mathbf{z}\rangle - \langle\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{W}}_L^{-\mathrm{T}}\cdot\mathbf{z},\mathbf{z}\rangle$
But

$$\langle\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{W}}_L^{-\mathrm{T}}\cdot\mathbf{z},\mathbf{z}\rangle \leq \langle\mathbf{z},\mathbf{z}\rangle \times \max_{\mathbf{t}\neq\mathbf{0}}\frac{\langle\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{W}}_L^{-\mathrm{T}}\cdot\mathbf{t},\mathbf{t}\rangle}{\langle\mathbf{t},\mathbf{t}\rangle} = \langle\mathbf{z},\mathbf{z}\rangle\times\lambda_{\max}(\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{W}}_L^{-\mathrm{T}}).$$

If $\lambda_{\max}(\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{W}}_L^{-\mathrm{T}}) \leq 2$ (which is satisfied for LORASC for example) then:

$$\langle\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{W}}_L^{-\mathrm{T}}\cdot\mathbf{z},\mathbf{z}\rangle \leq 2\langle\mathbf{z},\mathbf{z}\rangle;$$

Therefore, $\langle(2\mathbf{I}-\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{W}}_L^{-\mathrm{T}})\cdot\mathbf{z},\mathbf{z}\rangle \geq 0$ for all $\mathbf{z}\in\mathbb{R}^{n_L}$, in particular for $\mathbf{z} := \widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_{LR}\mathbf{W}_2^{-\mathrm{T}}\cdot\mathbf{y}$.
Thus, it follows from (3.21) that:

$$\langle\widehat{\mathbf{S}}\cdot\mathbf{y},\mathbf{y}\rangle \leq \langle\mathbf{W}_2^{-1}\mathbf{A}_R\mathbf{W}_2^{-\mathrm{T}}\cdot\mathbf{y},\mathbf{y}\rangle.$$

Hence:

$$\lambda_{\max}(\widehat{\mathbf{S}}) \leq \lambda_{\max}(\mathbf{W}_2^{-1}\mathbf{A}_R\mathbf{W}_2^{-\mathrm{T}}).$$

By injecting this latter inequality in (3.19) we obtain:

$$\lambda_{\max}(\widetilde{\mathbf{W}}^{-1}\mathbf{A}\widetilde{\mathbf{W}}^{-\mathrm{T}}) + \lambda_{\min}(\widetilde{\mathbf{W}}^{-1}\mathbf{A}\widetilde{\mathbf{W}}^{-\mathrm{T}}) \leq 2 + \lambda_{\max}(\mathbf{W}_2^{-1}\mathbf{A}_R\mathbf{W}_2^{-\mathrm{T}}) \tag{3.22}$$

By multiplying (3.22) by (3.20), we obtain Inequality (3.18). □

Remark 3.1. The bound (3.18) is valid when a LORASC preconditioner is used as an approximation of the L-block since it ensures that $\lambda_{\max}(\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{W}}_L^{-\mathrm{T}}) \leq 1$. Moreover, this bound can slightly be improved

$$\begin{aligned}&\kappa(\widetilde{\mathbf{W}}^{-1}\mathbf{A}\widetilde{\mathbf{W}}^{-\mathrm{T}}) + (\kappa(\widetilde{\mathbf{W}}^{-1}\mathbf{A}\widetilde{\mathbf{W}}^{-\mathrm{T}}))^{-1} + 2 \leq\\ &\Big(1+\lambda_{\max}(\mathbf{W}_2^{-1}\mathbf{A}_R\mathbf{W}_2^{-\mathrm{T}})\Big)\Big(\lambda_{\min}^{-1}(\mathbf{W}_2^{-1}\mathbf{S}\mathbf{W}_2^{-\mathrm{T}}) + \lambda_{\min}^{-1}(\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{W}}_L^{-\mathrm{T}})\\ &+\lambda_{\max}(\widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_{LR}\mathbf{S}^{-1}\mathbf{A}_{RL}\widetilde{\mathbf{W}}_L^{-\mathrm{T}})\times(\lambda_{\max}(\widetilde{\mathbf{W}}_L^{\mathrm{T}}\mathbf{A}_L^{-1}\widetilde{\mathbf{W}}_L)-1)^2\Big)\end{aligned} \tag{3.23}$$

When comparing the inequalities involving the condition number of the preconditioned operator with the exact and approximate adaptive preconditioners, we can see that the formulas are similar and have the same shapes. In fact, the left factor in the bound of (3.23) is equal to the one in (3.12). This is due to the fact that the low-rank approximation on the block $\mathbf{A}_L$ bounds the maximum eigenvalue of the corresponding block of the preconditioned matrix by 1. By contrast, the right factor in (3.23) contains more terms than that of (3.23). This is explained by the fact that the preconditioned operator does not have a block-diagonal shape when the approximate adaptive preconditioner is used, as is the case with the exact one.

3.5 Preconditioning cost

We devote this subsection to take a look at the cost associated with the use of an adaptive preconditioner. Furthermore, we propose a comparison between the preconditioning costs of exact and approximate adaptive preconditioners. Those include the build costs and the application costs but we will focus exclusively on the build costs as the application costs should not in principle differ much because the two preconditioners have the same shape.

Let $\mathbf{M}_2$ be an arbitrary SPD preconditioner of size n_R. We denote $\mathbf{M}_{\text{adap}}$ the adaptive preconditioner using exact inverse of $\mathbf{A}_L$ as defined in (3.3). As far as the build is concerned, constructing $\mathbf{M}_{\text{adap}}$ requires factorizing the matrix $\mathbf{A}_L$ whereas constructing $\widetilde{\mathbf{M}}_{\text{P}}(\widetilde{\mathbf{A}}_L, \mathbf{M}_2)$ requires factorizing a small diagonal block $\mathbf{A}^L_{\Gamma\Gamma} = \mathbf{C}_\Gamma \mathbf{C}_\Gamma^{\text{T}}$ of the block-arrow matrix $\mathbf{A}_L$ and computing a few smallest eigenvalues of the generalized eigenvalue problem

$$\mathbf{S}^L \cdot \mathbf{u} = \lambda \mathbf{A}^L_{\Gamma\Gamma} \cdot \mathbf{u}; \tag{3.24}$$

which is equivalent to the following eigenvalue problem:

$$\mathbf{C}_\Gamma^{-1}(\mathbf{A}^L_{\Gamma\Gamma} - \mathbf{S}^L)\mathbf{C}_\Gamma^{-\text{T}} \cdot \bar{\mathbf{u}} = \zeta \bar{\mathbf{u}} \tag{3.25}$$

where $\zeta = 1 - \lambda$ and $\bar{\mathbf{u}} = \mathbf{C}_\Gamma^{\text{T}}\mathbf{u}$. Using Cholesky factorization to transform a generalized eigenvalue problem involving symmetric matrices into a standard eigenvalue problem is common practice [1]. Note that both matrices $\mathbf{A}^L_{\Gamma\Gamma}$ and $\mathbf{S}^L$ in (3.24) are SPD therefore $\lambda > 0$. In addition, with Lemma 3.1, we get $0 < \lambda \leq 1$. Consequently, $0 \leq \zeta < 1$. This implies that the smallest eigenvalues of (3.24) and their associated eigenvectors can be deduced from the largest eigenvalues of (3.25) and the corresponding eigenvectors. This technique allows to avoid the costly computation of the smallest eigenvalues (and their eigenvectors). Indeed, the iterative process of computing the smallest eigenvalues is based on a shift and invert approach [124], which requires applying iteratively the operator $(\mathbf{S}^L - \mu\mathbf{A}^L_{\Gamma\Gamma})^{-1}\mathbf{A}^L_{\Gamma\Gamma}$, i.e. solving in each iteration a system with the matrix $\mathbf{S}^L - \mu\mathbf{A}^L_{\Gamma\Gamma}$, where μ here stands for a shift. This latter scalar is an initial approximate guess in the vicinity of which the smallest eigenvalues are located [124, Section 4.1.3]. This approach, called spectral transformation, can also be used for eigenvalues located in the interior of the spectrum [54, 68].

Furthermore, for the computation of few largest eigenvalues of the matrix $\mathbf{W}_{\Gamma\Gamma} := \mathbf{C}_\Gamma^{-1}(\mathbf{A}^L_{\Gamma\Gamma} - \mathbf{S}^L)\mathbf{C}_\Gamma^{-\text{T}})$, we apply the Implicitly Restarted Lanczos Method (IRLM [131]).The procedure can be broadly summarized as follows. Let n_{ev} be the number of eigenvalues desired. In ARPACK, this implies a Krylov space of dimension at least equal to $m = 2n_{\text{ev}}$ is going to be constructed [95, Section 2.3.3]. A Lanczos on this Krylov space would approximately cost $(2\text{nnz}(\mathbf{W}_{\Gamma\Gamma}) \times m)$ operations and yield an orthogonal basis $\mathbf{V}$ where the symmetric operator $\mathbf{W}_{\Gamma\Gamma}$ would be represented by the real symmetric tridiagonal matrix $\mathbf{H} := \mathbf{V}^{\text{T}}\mathbf{W}_{\Gamma\Gamma}\mathbf{V}$. The eigenpairs of this latter matrix are then computed, at a cost of $6m^2$ or $m\log(m)$ depending on the matrix and the method chosen, QR or Divide and Conquer, see [42, Section 4] and [24, 71, 39]. After that, the eigenvectors for the initial matrix $\mathbf{W}_{\Gamma\Gamma}$ are formed from those of $\mathbf{H}$ and the basis $\mathbf{V}$ at a cost of $m^2 \times n_\Gamma$, n_Γ being the size of $\mathbf{W}_{\Gamma\Gamma}$ (see [95, Section 2.3.3]. As the process is partially iterated (on the unwanted Ritz eigenpairs, see [132, Section 3]) a few times if not all the eigenvalues have converged, the total cost for finding the wanted eigenvalues would be estimated at $\mathcal{O}(2\text{nnz}(W_{\Gamma\Gamma}) \times m + 6m^2 + m^2 \times n_\Gamma)$ or

$\mathcal{O}(2\text{nnz}(W_{\Gamma\Gamma}) \times m + m\log(m) + m^2 \times n_\Gamma)$. According to the definition of $\mathbf{S}^L$ following (3.5), the matrix $\mathbf{W}_{\Gamma\Gamma}$ can be expressed:

$$\mathbf{W}_{\Gamma\Gamma} = \mathbf{C}_\Gamma^{-1}(\sum_{j=1}^{N} \mathbf{A}_{\Gamma j}^L (\mathbf{A}_{jj}^L)^{-1} \mathbf{A}_{j\Gamma}^L)\mathbf{C}_\Gamma^{-\mathrm{T}}.$$

As a consequence, $\mathbf{W}_{\Gamma\Gamma}$ is likely to be dense, $\text{nnz}(\mathbf{W}_{\Gamma\Gamma})$ is at most equal to n_Γ^2 and the Lanczos phase is the most dominant one in terms of complexity cost for the computation of few largest eigenpairs of $\mathbf{W}_{\Gamma\Gamma}$ with $\mathcal{O}(2n_\Gamma^2 \times m)$ operations at most. Building the exact adaptive preconditioner requires the exact inverse of $\mathbf{A}_L$ which is tantamount, in a way, to constructing $(\mathbf{S}^L)^{-1}$ (see (3.6)). The cost of this computation is $\mathcal{O}(n_\Gamma^3/3)$ operations. As we are interested in a few largest eigenvalues, the Krylov subspace dimension m is taken small with respect to the matrix size n_Γ. For this reason, the cost of the exact preconditioner $\mathcal{O}(n_\Gamma^3/3)$ is higher than that of the approximate one $\mathcal{O}(2n_\Gamma^2 \times m)$.

3.6 Possible choices for $\mathbf{M}_2$

From this point forward, the discussion that follows and the properties proven are only theoretical. For the numerical tests (Section 3.7), we deal with a block-Jacobi preconditioner $\mathbf{M}_2$ computed from $\mathbf{A}_R$ for comparison purposes with the numerical results of Chapter 2.

It is worth mentioning here that formulas similar to (3.9) and (3.10) were used and analyzed in some previous studies [109, 121, 129, 134]. In another article [13], this kind of preconditioners were studied in a broader way with arbitrary 2×2 splittings. In particular, we note that Formula (6) in [13, Section 4] is identical to the inverse of (3.9) whereas (3.10) corresponds to the particular case where only one approximate inverse is used ($\mathbf{M}_1 = \widetilde{\mathbf{A}}_L$) and was cited in [109, 129].

In this latter case, a good candidate for $\mathbf{M}_2$ can be deduced from expressing $\widetilde{\mathbf{M}}^{-1}\mathbf{A}$. Indeed, when $\mathbf{M}_2 = \mathbf{A}_R - \mathbf{A}_{RL}\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_{LR}$ then $\widetilde{\mathbf{M}}^{-1}\mathbf{A}$ has a block lower triangular shape and its lower right block corresponds to the identity matrix.

The question that may arise at this level is, what is the ideal preconditioner $\mathbf{M}_2$, when $\widetilde{\mathbf{A}}_L^{-1}$ the approximation of $\mathbf{A}_L^{-1}$ is fixed? Axelsson *et al.* answered this question and proved in [13, Section 4] that the ideal choice is

$$\begin{aligned}
&\mathbf{M}_2 = \alpha^{-1}\mathbf{D}_2;\\
&\mathbf{D}_2 := \mathbf{A}_R - \mathbf{A}_{RL}(\widetilde{\mathbf{A}}_L^{-1} + \widetilde{\mathbf{A}}_L^{-\mathrm{T}} - \widetilde{\mathbf{A}}_L^{-\mathrm{T}}\mathbf{A}_L\widetilde{\mathbf{A}}_L^{-1})\mathbf{A}_{LR}; \qquad (3.26)\\
&\alpha := \text{trace}(\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_L)/n_L. \qquad (3.27)
\end{aligned}$$

The formulas above determine the optimal preconditioner in the sense of trace/determinant ratio of the preconditioned matrix. This ratio is referred to as the K-condition number and it is an alternative quantity to the classical condition number that is used to obtain estimates of the number of PCG iterations [82]. The scalar α defined in (3.27) serves to minimize the K-condition number. As for the condition number $\kappa(\widetilde{\mathbf{M}}_\Gamma^{-1}\mathbf{A})$, it is sufficient to consider a block $\mathbf{M}_2$ that is proportional to $\mathbf{D}_2$, as detailed below. For ease of reference, we keep the notation used in that

article and denote $\mathbf{H}$ the global preconditioner $\widetilde{\mathbf{M}}_{\mathrm{P}}^{-1}$ constructed with $\mathbf{M}_2 = \mathbf{D}_2$ of (3.26):

$$\mathbf{H} := \widetilde{\mathbf{M}}_{\mathrm{P}}^{-1}(\widetilde{\mathbf{A}}_L, \mathbf{D}_2). \tag{3.28}$$

In what follows, and for a chosen approximation $\widetilde{\mathbf{A}}_L$, we further explain why the matrix $\mathbf{D}_2$ (that defines $\mathbf{H}$) is a good candidate as far as the minimization of the condition number of $\widetilde{\mathbf{M}}_{\mathrm{P}}^{-1}(\widetilde{\mathbf{A}}_L, \mathbf{M}_2)\mathbf{A}$ for a varying $\mathbf{M}_2$ is concerned. For that, we denote by γ the C.B.S. constant for the partitioning (3.1) and follow the same reasoning as in [12, Section 4]. Lemma 3.5 recalls a property of Theorem 9.5 of [12] that states that with a 2×2-block preconditioner built from exact off-diagonal blocks and approximations of the diagonal blocks of the matrix $\mathbf{A}$ can bound (from above and below) the eigenvalues of the preconditioned operator. The bounds of Lemma 3.5 are not optimal like the original ones of [12, Theorem 9.5] but their expressions are simpler, shorter and sufficient for the purpose of this section. For symmetric positive semi-definite matrices $\mathbf{B}_1$ and $\mathbf{B}_2$ in $\mathbb{R}^{n\times n}$, we recall the following notation: $\mathbf{B}_1 \leq \mathbf{B}_2$ if and only if $\forall \mathbf{y} \in \mathbb{R}^n, \mathbf{y}^{\mathrm{T}} \cdot \mathbf{B}_1 \cdot \mathbf{y} \leq \mathbf{y}^{\mathrm{T}} \cdot \mathbf{B}_2 \cdot \mathbf{y}$.

Lemma 3.5. *Let* $\mathbf{C} := \begin{pmatrix} \widetilde{\mathbf{A}}_L & \mathbf{A}_{LR} \\ \mathbf{A}_{RL} & \widetilde{\mathbf{A}}_R \end{pmatrix}$ *with* $\widetilde{\mathbf{A}}_L$ *and* $\widetilde{\mathbf{A}}_R$ *two approximations of* $\mathbf{A}_L$ *and* $\mathbf{A}_R$ *respectively such that there exist scalars* α_0, α_1, β_0 *and* β_1 *such that* $0 < \alpha_1 \leq 1 \leq \alpha_0$ *and* $0 < \beta_1 \leq 1 \leq \beta_0$ *and:*

$$\alpha_1 \mathbf{A}_L \leq \widetilde{\mathbf{A}}_L \leq \alpha_0 \mathbf{A}_L \tag{3.29}$$

$$\beta_1 \mathbf{A}_R \leq \widetilde{\mathbf{A}}_R \leq \beta_0 \mathbf{A}_R \tag{3.30}$$

Then

$$\lambda_{\max}(\mathbf{C}^{-1}\mathbf{A}) \leq \left(1 - \max\left\{\frac{1-\alpha_1}{1+\gamma}, \frac{1-\beta_1}{1+\gamma}\right\}\right)^{-1} \tag{3.31}$$

$$\lambda_{\min}(\mathbf{C}^{-1}\mathbf{A}) \geq \left(1 + \max\left\{\frac{\alpha_0-1}{1-\gamma}, \frac{\beta_0-1}{1-\gamma}\right\}\right)^{-1} \tag{3.32}$$

$$\mathcal{K}(\mathbf{C}^{-1}\mathbf{A}) \leq \left(1 + \max\left\{\frac{\alpha_0-1}{1-\gamma}, \frac{\beta_0-1}{1-\gamma}\right\}\right)\left(1 - \max\left\{\frac{1-\alpha_1}{1+\gamma}, \frac{1-\beta_1}{1+\gamma}\right\}\right)^{-1} \tag{3.33}$$

Proof. In a similar way to the proof of Theorem 9.5 in [12], with $\zeta_0 = 1, \zeta_1 = 1$. □

Remark 3.2. Given the matrices $\mathbf{A}_L$ and $\mathbf{A}_R$, note that inequalities like (3.29) and (3.30) can be obtained by choosing a preconditioner that bounds the extreme eigenvalues of the preconditioned operator. In such case, the scalars α_0 and α_1 can be respectively taken as the inverse of the lowerbound ($\frac{1}{\xi_0}$) and the inverse of the upperbound ($\frac{1}{\xi_1}$) of the preconditioned operator's eigenvalues as described in the following. Indeed, for the matrix $\mathbf{A}_L$ for example, let us denote $\widetilde{\mathbf{A}}_L = \widetilde{\mathbf{W}}_L \widetilde{\mathbf{W}}_L^{\mathrm{T}}$ the preconditioner that bounds the preconditioned operator's eigenvalues, i.e.:

$$\exists \xi_0 > 0,\ \exists \xi_1 > 0, \quad \xi_0 \leq \lambda(\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_L) \leq \xi_1$$

Then, by introducing the Rayleigh quotient for arbitrary non-zero vectors, we have:

$$\forall \mathbf{x} \neq \mathbf{0}, \quad \xi_0 \leq \lambda_{\min}(\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_L) \leq \frac{\mathbf{x}^{\mathrm{T}} \cdot \widetilde{\mathbf{W}}_L^{-1}\mathbf{A}_L\widetilde{\mathbf{W}}_L^{-\mathrm{T}} \cdot \mathbf{x}}{\mathbf{x}^{\mathrm{T}}\mathbf{x}} \leq \lambda_{\max}(\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_L) \leq \xi_1.$$

Therefore by considering $\mathbf{y} := \widetilde{\mathbf{W}}_L^{-\mathrm{T}} \cdot \mathbf{x}$, we get:

$$\forall \mathbf{y} \neq \mathbf{0}, \quad \xi_0(\mathbf{y}^{\mathrm{T}} \cdot \widetilde{\mathbf{W}}_L \widetilde{\mathbf{W}}_L^{\mathrm{T}} \cdot \mathbf{y}) \leq \mathbf{y}^{\mathrm{T}} \cdot \mathbf{A}_L \cdot \mathbf{y} \leq \xi_1(\mathbf{y}^{\mathrm{T}} \cdot \widetilde{\mathbf{W}}_L \widetilde{\mathbf{W}}_L^{\mathrm{T}} \cdot \mathbf{y}).$$

That is

$$\forall \mathbf{y} \neq \mathbf{0}, \quad \frac{1}{\xi_1}(\mathbf{y}^{\mathrm{T}} \cdot \mathbf{A}_L \cdot \mathbf{y}) \leq \mathbf{y}^{\mathrm{T}} \cdot \widetilde{\mathbf{A}}_L \cdot \mathbf{y} \leq \frac{1}{\xi_0}(\mathbf{y}^{\mathrm{T}} \cdot \mathbf{A}_L \cdot \mathbf{y}).$$

It may be inferred from Lemma 3.5, in particular from Inequality (3.33), that the closer to each other the scalars α_0, α_1 and β_0, β_1 respectively are, the smaller the upper bound on the condition number of $\mathbf{C}^{-1}\mathbf{A}$ is, and thereby the better the quality of the preconditioner $\mathbf{C}$ becomes. In other words, we may consider this latter as a function of the gaps ($\Delta_\alpha := \alpha_0 - \alpha_1$ and $\Delta_\beta := \beta_0 - \beta_1$) that is optimal when $(\Delta_\alpha, \Delta_\beta) = (0, 0)$.

Note that $\widetilde{\mathbf{M}}_{\mathrm{P}}(\widetilde{\mathbf{A}}_L, \mathbf{M}_2)$ of (3.10) is a particular case of $\mathbf{C}$ in Lemma 3.5. In the sequel, we will see how this latter applies to $\widetilde{\mathbf{M}}_{\mathrm{P}}(\widetilde{\mathbf{A}}_L, \mathbf{M}_2)$ with some choices of $\mathbf{M}_2$.

Corollary 3.1. *Let $\widetilde{\mathbf{A}}_L$ and $\mathbf{M}_2$ be such that there exist scalars α_0, α_1, θ_0 and θ_1 such that $0 < \alpha_1 \leq 1 \leq \alpha_0$ and $0 < \theta_1 \leq 1 \leq \theta_0$ and:*

$$\alpha_1 \mathbf{A}_L \leq \widetilde{\mathbf{A}}_L \leq \alpha_0 \mathbf{A}_L \tag{3.34}$$

$$\theta_1 \mathbf{A}_R \leq \mathbf{M}_2 \leq \theta_0 \mathbf{A}_R \tag{3.35}$$

Then

$$\mathcal{K}(\widetilde{\mathbf{M}}_{\mathrm{P}}^{-1}(\widetilde{\mathbf{A}}_L, \mathbf{M}_2)\mathbf{A}) \leq \left(1 + \max\left\{\frac{\alpha_0 - 1}{1-\gamma}, \frac{\theta_0 + \alpha_1^{-1}\gamma^2 - 1}{1-\gamma}\right\}\right)\left(1 - \max\left\{\frac{1-\alpha_1}{1+\gamma}, \frac{1-\theta_1}{1+\gamma}\right\}\right)^{-1} \tag{3.36}$$

Proof. We know that for all vectors $\mathbf{x}$ and $\mathbf{y}$ of lengths n_L and n_R respectively, we have:

$$\frac{(\mathbf{x}^{\mathrm{T}} \cdot \mathbf{A}_{LR} \cdot \mathbf{y})^2}{\mathbf{x}^{\mathrm{T}} \cdot \mathbf{A}_L \cdot \mathbf{x}} \leq \gamma^2 \mathbf{y}^{\mathrm{T}} \cdot \mathbf{A}_R \cdot \mathbf{y}$$

By taking $\mathbf{x} := \mathbf{A}_L^{-1}\mathbf{A}_{LR} \cdot \mathbf{y}$ for arbitrary $\mathbf{y}$, we get:

$$\frac{(\mathbf{y}^{\mathrm{T}} \cdot \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR} \cdot \mathbf{y})^2}{\mathbf{y}^{\mathrm{T}} \cdot \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_L\mathbf{A}_L^{-1}\mathbf{A}_{LR} \cdot \mathbf{y}} = \mathbf{y}^{\mathrm{T}} \cdot \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR} \cdot \mathbf{y} \leq \gamma^2 \mathbf{y}^{\mathrm{T}} \cdot \mathbf{A}_R \cdot \mathbf{y}$$

This means that $\mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR} \leq \gamma^2 \mathbf{A}_R$.
Note here that in the case when $n_L < n_R$, the matrix $\mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR}$ is singular and thus there exists no positive scalar ω such that: $\omega \mathbf{A}_R \leq \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR}$. As a consequence, we can only write:

$$\mathbf{0} \leq \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR} \leq \gamma^2 \mathbf{A}_R \tag{3.37}$$

From (3.34), we obtain:

$$\alpha_0^{-1}\mathbf{A}_L^{-1} \leq \widetilde{\mathbf{A}}_L^{-1} \leq \alpha_1^{-1}\mathbf{A}_L^{-1}$$

Hence

$$\alpha_0^{-1}\mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR} \leq \mathbf{A}_{RL}\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_{LR} \leq \alpha_1^{-1}\mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR}$$

After injecting (3.37), we obtain:

$$\mathbf{0} \leq \mathbf{A}_{RL}\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_{LR} \leq \alpha_1^{-1}\gamma^2\mathbf{A}_R$$

Thus

$$\theta_1\mathbf{A}_R \leq \mathbf{M}_2 + \mathbf{A}_{RL}\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_{LR} \leq (\theta_0 + \alpha_1^{-1}\gamma^2)\mathbf{A}_R$$

Finally, all that remains is to apply Lemma 3.5 with $\mathbf{C} = \widetilde{\mathbf{M}}_{\mathrm{P}}(\widetilde{\mathbf{A}}_L, \mathbf{M}_2)$, $\beta_0 = \theta_0 + \alpha_1^{-1}\gamma^2$ and $\beta_1 = \theta_1$. □

Corollary 3.2. *Let $\widetilde{\mathbf{A}}_L$ be such that there exist scalars α_0, α_1:*

$$\gamma^2(1-\alpha_0^{-1}) < \alpha_1 \leq 1 \leq \alpha_0 \tag{3.38}$$

$$\alpha_1\mathbf{A}_L \leq \widetilde{\mathbf{A}}_L \leq \alpha_0\mathbf{A}_L \tag{3.39}$$

Then

$$\mathcal{K}(\widetilde{\mathbf{M}}_{\mathrm{P}}^{-1}(\widetilde{\mathbf{A}}_L, \mathbf{D}_2)\mathbf{A}) \leq \left(1 + \max\left\{\frac{\alpha_0 - 1}{1-\gamma}, \frac{(1-\alpha_1)\gamma^2}{(1-\gamma)\alpha_1^2}\right\}\right)\left(1 - \max\left\{\frac{1-\alpha_1}{1-\gamma}, \frac{\gamma^2(\alpha_0-1)}{\alpha_0\alpha_1(1-\gamma)}\right\}\right)^{-1} \tag{3.40}$$

Proof. The bottom right diagonal block of $\widetilde{\mathbf{M}}_{\mathrm{P}}(\widetilde{\mathbf{A}}_L, \mathbf{D}_2)$ equals to

$$\mathbf{D}_2 + \mathbf{A}_{RL}\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_{LR} = \mathbf{A}_R - \mathbf{A}_{RL}(\widetilde{\mathbf{A}}_L^{-\mathrm{T}} - \widetilde{\mathbf{A}}_L^{-\mathrm{T}}\mathbf{A}_L\widetilde{\mathbf{A}}_L^{-1})\mathbf{A}_{LR}\ .$$

Due to (3.39), we have:

$$\alpha_0^{-1}\widetilde{\mathbf{A}}_L \leq \mathbf{A}_L \leq \alpha_1^{-1}\widetilde{\mathbf{A}}_L \qquad \text{and} \qquad \alpha_0^{-1}\mathbf{A}_L^{-1} \leq \widetilde{\mathbf{A}}_L^{-\mathrm{T}} \leq \alpha_1^{-1}\mathbf{A}_L^{-1}$$

Thus, we can write

$$\alpha_0^{-1}\widetilde{\mathbf{A}}_L^{-\mathrm{T}} \leq \widetilde{\mathbf{A}}_L^{-\mathrm{T}}\mathbf{A}_L\widetilde{\mathbf{A}}_L^{-1} \leq \alpha_1^{-1}\widetilde{\mathbf{A}}_L^{-\mathrm{T}}$$

Consequently, we have

$$\alpha_1^{-1}(1-\alpha_1^{-1})\mathbf{A}_L^{-1} \leq (1-\alpha_1^{-1})\widetilde{\mathbf{A}}_L^{-\mathrm{T}} \leq \widetilde{\mathbf{A}}_L^{-\mathrm{T}} - \widetilde{\mathbf{A}}_L^{-\mathrm{T}}\mathbf{A}_L\widetilde{\mathbf{A}}_L^{-1} \leq (1-\alpha_0^{-1})\widetilde{\mathbf{A}}_L^{-\mathrm{T}} \leq \alpha_1^{-1}(1-\alpha_0^{-1})\mathbf{A}_L^{-1}$$

Then

$$\alpha_1^{-1}(\alpha_0^{-1}-1)\mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR} \leq -\mathbf{A}_{RL}(\widetilde{\mathbf{A}}_L^{-\mathrm{T}} - \widetilde{\mathbf{A}}_L^{-\mathrm{T}}\mathbf{A}_L\widetilde{\mathbf{A}}_L^{-1})\mathbf{A}_{LR} \leq \alpha_1^{-1}(\alpha_1^{-1}-1)\mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR}$$

And with (3.37) we obtain:

$$\gamma^2\alpha_1^{-1}(\alpha_0^{-1}-1)\mathbf{A}_R \leq -\mathbf{A}_{RL}(\widetilde{\mathbf{A}}_L^{-\mathrm{T}} - \widetilde{\mathbf{A}}_L^{-\mathrm{T}}\mathbf{A}_L\widetilde{\mathbf{A}}_L^{-1})\mathbf{A}_{LR} \leq \gamma^2\alpha_1^{-1}(\alpha_1^{-1}-1)\mathbf{A}_R$$

Therefore

$$\left(1+\gamma^2\alpha_1^{-1}(\alpha_0^{-1}-1)\right)\mathbf{A}_R \leq \mathbf{A}_R - \mathbf{A}_{RL}(\widetilde{\mathbf{A}}_L^{-\mathrm{T}} - \widetilde{\mathbf{A}}_L^{-\mathrm{T}}\mathbf{A}_L\widetilde{\mathbf{A}}_L^{-1})\mathbf{A}_{LR} \leq \left(1+\gamma^2\alpha_1^{-1}(\alpha_1^{-1}-1)\right)\mathbf{A}_R$$

To complete the proof, we apply Lemma 3.5 to $\widetilde{\mathbf{A}}_L$, $\widetilde{\mathbf{A}}_R := \mathbf{D}_2 + \mathbf{A}_{RL}\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_{LR}$, $\beta_0 := 1+\gamma^2\alpha_1^{-1}(\alpha_1^{-1}-1)$ and $\beta_1 := 1+\gamma^2\alpha_1^{-1}(\alpha_0^{-1}-1)$. Thanks to (3.38), this latter coefficient is positive. □

Remark 3.3. We shall make some remarks before we move to the next theorem.

— In the case of a general $\mathbf{M}_2$, Corollary 3.1 gives the formulas for the scalars β_0 and β_1, as defined in Lemma 3.5, that are involved in the bounds of the extreme eigenvalues of $\widetilde{\mathbf{M}}_{\mathrm{P}}^{-1}(\widetilde{\mathbf{A}}_L, \mathbf{M}_2)\mathbf{A}$:

$$\beta_0 := \theta_0 + \alpha_1^{-1}\gamma^2; \qquad \beta_1 := \theta_1;$$

thus yielding the gap $\Delta_\beta = \theta_0 - \theta_1 + \alpha_1^{-1}\gamma^2 \geq \gamma^2$. This means that even if we improve the quality of the approximations $\widetilde{\mathbf{A}}_L$ and $\mathbf{M}_2$ by respectively reducing the gaps Δ_α and $\Delta_\theta := \theta_0 - \theta_1$, we do not necessarily reduce the gap Δ_β. Indeed, this latter is bounded from below by a constant that does not depend on the approximations.

— In the particular case of $\mathbf{D}_2$, Corollary 3.2 gives the formulas for the scalars β_0 and β_1, as defined in Lemma 3.5, that are involved in the bounds of the extreme eigenvalues of $\widetilde{\mathbf{M}}_{\mathrm{P}}^{-1}(\widetilde{\mathbf{A}}_L, \mathbf{D}_2)\mathbf{A}$:

$$\beta_0 := 1+\gamma^2\alpha_1^{-1}(\alpha_1^{-1}-1); \qquad \beta_1 := 1+\gamma^2\alpha_1^{-1}(\alpha_0^{-1}-1);$$

Thus yielding the gap $\Delta_\beta = \gamma^2\alpha_1^{-1}(\alpha_1^{-1}-\alpha_0^{-1}) = \dfrac{\gamma^2}{\alpha_0\alpha_1^2} \times \Delta_\alpha$. As a consequence, improving the quality of the approximation $\widetilde{\mathbf{A}}_L$ results not only in a lower Δ_α but also in a lower Δ_β because those two quantities are proportional. Hence, the relevance of the choice $\mathbf{M}_2 = \mathbf{D}_2$ for a low condition number of $\widetilde{\mathbf{M}}_{\mathrm{P}}^{-1}(\widetilde{\mathbf{A}}_L, \mathbf{M}_2)\mathbf{A}$.

In spite of the good theoretical property of $\mathbf{D}_2$, the question whether this matrix is computable in practice remains open. If so, applying the preconditioner $\widetilde{\mathbf{M}}_{\mathrm{P}}^{-1}(\widetilde{\mathbf{A}}_L, \mathbf{D}_2)$ requires either inverting the matrix $\mathbf{D}_2$ or applying $\mathbf{D}_2^{-1}$ to vectors of size n_R. Yet, it may not be feasible in practice to compute or apply its inverse to arbitrary vectors especially if it is too dense. For such cases, we propose the alternative preconditioning strategy that is presented in the following theorem.

Theorem 3.1. *Let $\widetilde{\mathbf{A}}_L^{-1}$ be a given approximation of $\mathbf{A}_L^{-1}$, close enough so that $\mathbf{D}_2$ is positive definite. We set*

$$\mathbf{M}^{-1} := \widetilde{\mathbf{M}}_{\mathrm{P}}^{-1}(\widetilde{\mathbf{A}}_L, \widetilde{\mathbf{D}}_2) = \begin{pmatrix} \mathbf{I} & -\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_{LR} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \widetilde{\mathbf{A}}_L^{-1} & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{D}}_2^{-1} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{A}_{RL}\widetilde{\mathbf{A}}_L^{-1} & \mathbf{I} \end{pmatrix}; \tag{3.41}$$

where $\widetilde{\mathbf{D}}_2$ is a close approximation of $\mathbf{D}_2$ that bounds the condition number of $\mathbf{D}_2\widetilde{\mathbf{D}}_2^{-1}$, by a predefined constant i.e.:

$$\exists C_1 > 0 \quad \textit{such that} \quad \kappa(\mathbf{D}_2\widetilde{\mathbf{D}}_2^{-1}) < C_1 \tag{3.42}$$

Then the preconditioner $\mathbf{M}$ *bounds the condition number of the preconditioned operator* $\mathbf{M}^{-1}\mathbf{A}$*:*

$$\exists C > 0 \quad \textit{such that} \quad \kappa(\mathbf{M}^{-1}\mathbf{A}) \leq C\, \kappa(\widetilde{\mathbf{M}}_{\mathrm{P}}^{-1}(\widetilde{\mathbf{A}}_L, \mathbf{D}_2)\mathbf{A}) \tag{3.43}$$

Proof. Let $\mathbf{L}_1$ and $\mathbf{L}$ such that $\mathbf{L}_1\mathbf{L}_1^{\mathrm{T}} = \mathbf{H}^{-1}$ and $\mathbf{L}^{-\mathrm{T}}\mathbf{L}^{-1} = \mathbf{M}^{-1}$ We have:

$$\begin{aligned}
\lambda_{\max}(\mathbf{M}^{-1}\mathbf{A}) = \lambda_{\max}(\mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{-\mathrm{T}}) &= \max_{\mathbf{y}\neq 0} \frac{\langle y, \mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{-\mathrm{T}}\cdot y\rangle}{\langle y, y\rangle} \\
&= \max_{\mathbf{y}\neq 0} \frac{\langle y, \mathbf{L}^{-1}\mathbf{L}_1\mathbf{L}_1^{-1}\mathbf{A}\mathbf{L}_1^{-\mathrm{T}}\mathbf{L}_1^{\mathrm{T}}\mathbf{L}^{-\mathrm{T}}\cdot y\rangle}{\langle y, y\rangle} \\
&= \max_{\mathbf{y}\neq 0} \frac{\langle \mathbf{L}_1^{\mathrm{T}}\mathbf{L}^{-\mathrm{T}}\cdot y, (\mathbf{L}_1^{-1}\mathbf{A}\mathbf{L}_1^{-\mathrm{T}})\cdot \mathbf{L}_1^{\mathrm{T}}\mathbf{L}^{-\mathrm{T}}\cdot y\rangle}{\langle \mathbf{L}_1^{\mathrm{T}}\mathbf{L}^{-\mathrm{T}}\cdot y, \mathbf{L}_1^{\mathrm{T}}\mathbf{L}^{-\mathrm{T}}\cdot y\rangle} \times \frac{\langle \mathbf{L}_1^{\mathrm{T}}\mathbf{L}^{-\mathrm{T}}\cdot y, \mathbf{L}_1^{\mathrm{T}}\mathbf{L}^{-\mathrm{T}}\cdot y\rangle}{\langle y, y\rangle} \\
&\leq \left(\max_{\mathbf{z}\neq 0} \frac{\langle z, \mathbf{L}_1^{-1}\mathbf{A}\mathbf{L}_1^{-\mathrm{T}}\cdot z\rangle}{\langle z, z\rangle}\right) \times \left(\max_{\mathbf{y}\neq 0} \frac{\langle y, \mathbf{L}^{-1}\mathbf{L}_1\mathbf{L}_1^{\mathrm{T}}\mathbf{L}^{-\mathrm{T}}\cdot y\rangle}{\langle y, y\rangle}\right) \\
&\leq \lambda_{\max}(\mathbf{L}_1^{-1}\mathbf{A}\mathbf{L}_1^{-\mathrm{T}}) \times \lambda_{\max}(\mathbf{L}^{-1}\mathbf{L}_1\mathbf{L}_1^{\mathrm{T}}\mathbf{L}^{-\mathrm{T}}) = \lambda_{\max}(\mathbf{H}\mathbf{A}) \times \lambda_{\max}(\mathbf{H}^{-1}\mathbf{M}^{-1})
\end{aligned}$$

In a similar fashion, we can show that

$$\lambda_{\min}(\mathbf{M}^{-1}\mathbf{A}) \geq \lambda_{\min}(\mathbf{H}\mathbf{A}) \times \lambda_{\min}(\mathbf{H}^{-1}\mathbf{M}^{-1});$$

and thus:

$$\kappa(\mathbf{M}^{-1}\mathbf{A}) = \frac{\lambda_{\max}(\mathbf{M}^{-1}\mathbf{A})}{\lambda_{\min}(\mathbf{M}^{-1}\mathbf{A})} \leq \kappa(\mathbf{H}\mathbf{A}) \times \kappa(\mathbf{H}^{-1}\mathbf{M}^{-1}) \tag{3.44}$$

From the definition of $\mathbf{H}$, we have:

$$\mathbf{H} = \begin{pmatrix} \mathbf{I} & -\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_{LR} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \widetilde{\mathbf{A}}_L^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_2^{-1} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{A}_{RL}\widetilde{\mathbf{A}}_L^{-1} & \mathbf{I} \end{pmatrix}$$

Therefore

$$\mathbf{H}^{-1} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{A}_{RL}\widetilde{\mathbf{A}}_L^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \widetilde{\mathbf{A}}_L & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_2 \end{pmatrix} \begin{pmatrix} \mathbf{I} & \widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_{LR} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}$$

And

$$\mathbf{H}^{-1}\mathbf{M}^{-1} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{A}_{RL}\widetilde{\mathbf{A}}_L^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_2\widetilde{\mathbf{D}}_2^{-1} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{A}_{RL}\widetilde{\mathbf{A}}_L^{-1} & \mathbf{I} \end{pmatrix}$$

Then: $\mathrm{Sp}(\mathbf{H}^{-1}\mathbf{M}^{-1}) = \{1\} \cup \mathrm{Sp}(\mathbf{D}_2\widetilde{\mathbf{D}}_2^{-1})$
From this latter equality and the fact that the condition number of $\mathbf{D}_2\widetilde{\mathbf{D}}_2^{-1}$ is bounded by C_1, we deduce that the condition number of $\mathbf{H}^{-1}\mathbf{M}^{-1}$ is bounded as well by a constant C that depends on C_1:

$$\exists C > 0 \quad \text{such that} \quad \kappa(\mathbf{H}^{-1}\mathbf{M}^{-1}) \leq C. \tag{3.45}$$

On the other hand, we have

$$\kappa(\mathbf{HA}) = \kappa(\widetilde{\mathbf{M}}_{\mathrm{P}}^{-1}(\widetilde{\mathbf{A}}_L, \mathbf{D}_2)\mathbf{A}). \tag{3.46}$$

Ultimately, Inequality (3.43) follows from (3.44), (3.46) and (3.45) □

There exist some preconditioners that allow to bound the condition number of the preconditioned operator, such as domain decomposition-based preconditioners (like BDD or two-level Schwarz) [47] or LORASC [67]. The first mentioned are not well suited in this context for $\widetilde{\mathbf{D}}_2$ because they usually require solution of subdomain problems arising from the discretization of the PDE. Yet, $\mathbf{D}_2$ does not stem from any PDE. On the other hand, LORASC preconditioner could be used to build $\widetilde{\mathbf{D}}_2$ as this preconditioner is fully algebraic and requires no other information other than the matrix. However, constructing this preconditioner for $\mathbf{D}_2$ requires finding a permutation of this latter matrix into a block-arrow shape and inverting exactly the diagonal blocks of the permuted matrix. This operation might be costly if we cannot guarantee that the size of the blocks in the block-arrow shape is small enough. The subsequent corollary presents a modified low rank approximation strategy as an alternative to the standard LORASC strategy.

Corollary 3.3. *Let $\widetilde{\mathbf{A}}_L^{-1}$ be a given approximation of $\mathbf{A}_L^{-1}$, close enough so that $\mathbf{D}_2$ is positive definite. Let $\mathbf{N}_2$ be a symmetric positive definite preconditioner of $\mathbf{D}_2$ that bounds the maximum eigenvalue of $\mathbf{N}_2^{-1}\mathbf{D}_2$:*

$$\exists C_2 > 0 \quad \textit{such that} \quad \lambda_{\max}(\mathbf{N}_2^{-1}\mathbf{D}_2) \leq C_2. \tag{3.47}$$

Let $0 < \epsilon < 1$ be a given threshold, and $\lambda_1, \ldots, \lambda_{n_R}$ be the eigenvalues of $\mathbf{N}_2^{-1}\mathbf{D}_2$ sorted in an increasing order and $\mathbf{v}_1, \ldots, \mathbf{v}_{n_R}$ be the associated $\mathbf{N}_2$-orthonormal eigenvectors. Let also $i \in [\![1, n_R]\!]$ such that $\lambda_j < \epsilon \ , \quad \forall j \leq i$. We define $\mathbf{E}_i := [\mathbf{v}_1 \mathbf{v}_2 \ldots \mathbf{v}_i]$ and $\mathbf{\Sigma}_i = [\sigma_1 \sigma_2 \ldots \sigma_i]$ with $\sigma_j = \dfrac{\epsilon - \lambda_j}{\lambda_j} \ , \quad \forall j \leq i$. For the preconditioner

$$\mathbf{M}^{-1} := \widetilde{\mathbf{M}}_{\mathrm{P}}^{-1}(\widetilde{\mathbf{A}}_L, \widetilde{\mathbf{D}}_2) = \begin{pmatrix} \mathbf{I} & -\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_{LR} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \widetilde{\mathbf{A}}_L^{-1} & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{D}}_2^{-1} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{A}_{RL}\widetilde{\mathbf{A}}_L^{-1} & \mathbf{I} \end{pmatrix}; \tag{3.48}$$

where $\widetilde{\mathbf{D}}_2^{-1} := \mathbf{N}_2^{-1} + \mathbf{E}_i \mathbf{\Sigma}_i \mathbf{E}_i^{\mathrm{T}}$, we have:

$$\kappa(\mathbf{M}^{-1}\mathbf{A}) \leq \frac{C_2}{\epsilon} \kappa(\widetilde{\mathbf{M}}_{\mathrm{P}}^{-1}(\widetilde{\mathbf{A}}_L, \mathbf{D}_2)\mathbf{A}). \tag{3.49}$$

Proof. Following the same reasoning as in the proof of Theorem 3.1, we obtain:

$$\kappa(\mathbf{M}^{-1}\mathbf{A}) \leq \kappa(\mathbf{H}^{-1}\mathbf{M}^{-1}) \times \kappa(\widetilde{\mathbf{M}}_{\mathrm{P}}^{-1}(\widetilde{\mathbf{A}}_L, \mathbf{D}_2)\mathbf{A}). \tag{3.50}$$

Since $\mathrm{Sp}(\mathbf{H}^{-1}\mathbf{M}^{-1}) = \{1\} \cup \mathrm{Sp}(\widetilde{\mathbf{D}}_2^{-1}\mathbf{D}_2)$, we have:

$$\kappa(\mathbf{H}^{-1}\mathbf{M}^{-1}) = \kappa(\widetilde{\mathbf{D}}_2^{-1}\mathbf{D}_2).$$

On the other hand, and in a similar way to the proofs of Lemmas 3.1 and 3.2, we can show that:

$$\mathrm{Sp}(\widetilde{\mathbf{D}}_2^{-1}\mathbf{D}_2) = \{\epsilon, \lambda_{i+1}, \ldots, \lambda_{n_R}\}$$

Consequently,

$$\kappa(\mathbf{H}^{-1}\mathbf{M}^{-1}) = \frac{\lambda_{n_R}}{\epsilon} .$$

But,

$$\lambda_{n_R} = \lambda_{\max}(\mathbf{N}_2^{-1}\mathbf{D}_2) \leq C_2$$

Therefore

$$\kappa(\mathbf{M}^{-1}\mathbf{A}) \leq \frac{C_2}{\epsilon} \times \kappa(\widetilde{\mathbf{M}}_{\mathrm{P}}^{-1}(\widetilde{\mathbf{A}}_L, \mathbf{D}_2)\mathbf{A}).$$

□

Over the classical LORASC method, Corollary 3.3 presents the advantage of replacing the exact inverse of the interface submatrix by an approximate inverse. The alternative method is less cumbersome and requires only that the maximum eigenvalue be bound. This can be easily achieved by just taking a simple Block-Jacobi preconditioner for instance for $\mathbf{N}_2$.

3.7 Numerical results

In the previous sections, we explained the concept and motivations for the adaptive LORASC error-based preconditioner where the LORASC preconditioner is used as an approximate inverse that replaces the exact inverse of the submatrix $\mathbf{A}_L$ in the expression of the exact adaptive error-based preconditioner (see (3.3)). In this section, we analyze the efficiency of this new preconditioner on a set of matrices stemming from the discretization by the finite element method of diffusion models with inhomogeneous coefficients on two-dimensional (2D) domains. The following numerical experiments are based on Matlab with PDE toolbox in order to create a mesh and solve the considered PDE on a domain Ω. We use the 'initmesh' command with the maximum element size indicated by the input $H_{\max}$ to build a mesh according to Delaunay triangulation. Once the main linear system is defined after the discretization with $\mathbb{P}_1$ finite elements, we run 20 iterations of PCG to get a starting distribution of the algebraic error on all the elements of the mesh from which we distinguish subdomains L and R with significant high algebraic error and low algebraic error respectively. The interested reader can find in the previous chapter (Section 2.6.1) full details on the manner in which this domain splitting is carried out. Then the solve process is continued either with the initial preconditioner, chosen to be a Block-Jacobi preconditioner composed of 50 blocks, or an adaptive error-based preconditioner until the residual norm goes below the threshold 10^{-6}. In the sequel, we test and compare three adaptive preconditioners:

— The exact adaptive preconditioner which uses an exact factorization of $\mathbf{A}_L$.

— The approximate LORASC-based adaptive preconditioner which uses a low rank factorization to approximate $\mathbf{A}_L^{-1}$, see (3.10).

— The approximate IC(0)-adaptive preconditioner which uses an incomplete Cholesky factorization with zero-fill to approximate $\mathbf{A}_L^{-1}$. It is used as an example of incomplete-factorization-based preconditioner and serves merely for comparison purposes.

To construct the approximate LORASC-based adaptive preconditioner, the graph partitioner METIS [83] is used to obtain the block-arrow shape form of (3.4), and the function 'eigs' of Matlab, which uses ARPACK [95], is called to compute the smallest eigenvalues and associated eigenvectors of the generalized eigenvalues problem (3.24). The construction of LORASC is such that it deflates the small eigenvalues below a threshold $\epsilon = 0.2$, and thus the eigenvalues of the preconditioned matrix $\widetilde{\mathbf{A}}_L^{-1}\mathbf{A}_L$ are lower bounded by $\epsilon = 0.2$.

Let $\Omega \subset \mathbb{R}^2$ be a polytopal domain (open, bounded and connected set). The diffusion model with inhomogeneous coefficients that we consider here may be written as follows:

$$\begin{cases} -\nabla \cdot (\underline{\mathbf{K}} \nabla \underline{u}) = \underline{f} & \text{in } \Omega, \\ \underline{u} = \underline{u}_0 & \text{on } \partial\Omega, \end{cases} \tag{3.51}$$

where $\underline{u} : \Omega \to \mathbb{R}$ is the unknown function, $\underline{u}_0 : \partial\Omega \to \mathbb{R}$ is the Dirichlet boundary function, $\underline{f} : \Omega \to \mathbb{R}$ is a source term, and $\underline{\mathbf{K}}$ is an uniformly bounded and positive definite diffusion tensor. The diffusion tensor is defined as a multiple of the identity matrix $\underline{\mathbf{K}} = c * \mathbf{I}$. The factor c here stands for the diffusivity and can be a highly heterogeneous function of Ω. We consider in the sequel three test cases with different diffusivity functions. For the initial Block-Jacobi preconditioner and the three adaptive preconditioners derived at the beginning of this section, we plot curves that show the evolution of the global energy norm of the error and of the local portion of error captured in subset L. This latter quantity is the L-norm defined in Section 2.6.1 of Chapter 2. Furthermore, we study the effectiveness of the approximate adaptive preconditioners by examining their behavior in terms of weak and strong scalibility. We recall that the size of the problem proportionally increases with respect to the number of partitions in weak scaling while it is fixed and only the number of partitions varies in strong scaling.

3.7.1 Test case n°1

In the first test, we set the domain $\Omega = [-1, 1]$, and consider an homogeneous Dirichlet boundary condition on $\partial\Omega$. We fix the diffusivity to the value 1 on the whole domain and hence obtain a classical Poisson's equation. The source term f is appropriately chosen such that the exact solution of the PDE is:

$$\begin{aligned} u = {} & (x+1) \times (x-1) \times (y+1) \times (y-1) \times \\ & (\exp(-\alpha \times ((x+0.5)^2 + (y+0.5)^2) - \exp(-\beta \times ((x-0.5)^2 + (y-0.5)^2))), \end{aligned}$$

with $\alpha = 300$ and $\beta = 200$. The first test case of this chapter is similar to the second one of Chapter 2 where the error is localized on two circular spots of the domain. The parameters (α, β and the mesh size) were modified to increase the size of the set L. Table 3.1 gathers the number of iterations required to solve the preconditioned system (it_{LORASC} when adaptive LORASC-based preconditioner is considered, $it_{\text{IC}(0)}$ for adaptive preconditioner based on Incomplete Cholesky without fill-in, it_{exact} for exact adaptive preconditioner of (3.3) and it_{BJ} for the Block-Jacobi preconditioner) along with n_{ev} the number of deflated eigenvalues, N_{mult} the number of matrix-vector operations required to compute those eigenvalues via function 'eigs' of Matlab, and the dimension m of the search Krylov subspace, with respect to the problem size and the number of partitions N. The values of the condition number of the matrix $\mathbf{A}_L$ are reported in the table as

well ($\kappa(\mathbf{A}_L)$).

Table 3.1 – *Weak scaling results for test case n°1. The number of iterations are given for exact and approximate adaptive preconditioner using LORASC or Incomplete Cholesky IC(0).*

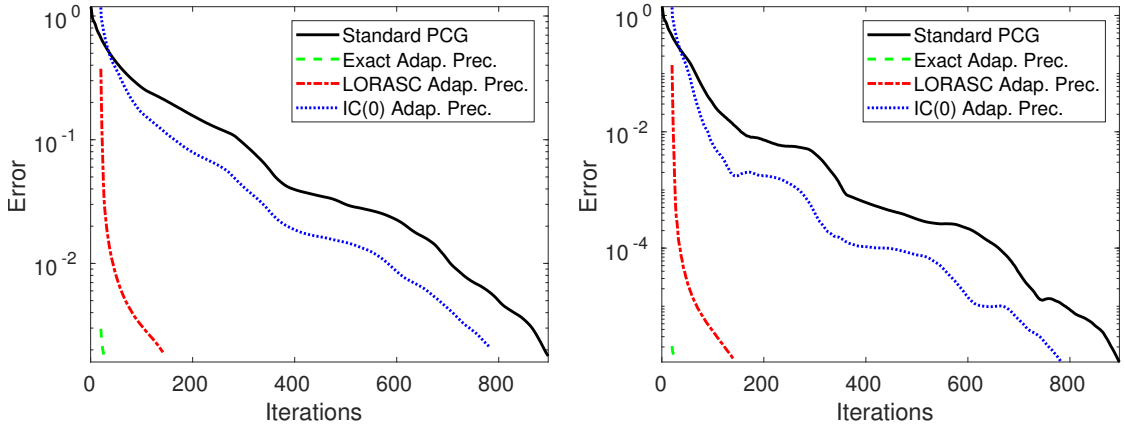
n	nnz	n_L	$\kappa(\mathbf{A}_L)$	N	n_Γ	$nnz(\mathbf{W}_\Gamma)$	m	n_{ev}	N_{mult}	it_{LORASC}	$it_{IC(0)}$	it_{exact}	it_{BJ}
68 280	494 203	10 244	3 898	8	680	190 072	90	21	152	192	463	87	537
107 649	494 203	16 149	6 233	16	1 435	734 343	190	44	310	162	568	37	666
191 183	494 203	28 672	11 836	32	2 976	2 921 064	396	92	631	123	763	10	897

Table 3.2 contains the results for the strong scaling. It outlines the changes in the dimension of search space, the number of eigenvalues extracted and iterations as the number of partitions increases.

Table 3.2 – *Strong scaling results for test case n°1.*

n	nnz	n_L	N	n_Γ	$nnz(\mathbf{W}_\Gamma)$	m	n_{ev}	N_{mult}	it_{LORASC}
191 183	1 334 875	28 672	4	651	209 853	86	21	146	170
191 183	1 334 875	28 672	8	1 183	577 748	156	40	256	109
191 183	1 334 875	28 672	16	1 935	1 337 690	258	60	415	128
191 183	1 334 875	28 672	32	2 976	2 921 064	396	92	631	123

Figure 3.1 depicts for the test case n°1 the way in which the error norms, the global error norm and the local portion of error in L, vary during the iterative solve with the four different preconditioners introduced earlier in this section. The matrix size and parameters' settings considered to plot these curves are the ones reported in the last row of Table 3.1.



(a) Energy norm of the algebraic error *(b) L-norm of the algebraic error*

Figure 3.1 – *Error evolution for test case n°2*

We remark from those Tables 3.1 and 3.2 that in general, only few eigenvalues need to be computed for LORASC. We also notice that n_{ev} and m grow when we increase the number of partitions. In addition, we underline the fact that the dimension of the search Krylov subspace considered here

always satisfies the condition required in [95]: $m \geq 2 * n_{\text{ev}}$. In addition, we may infer from the results in Table 3.4 that the number of partitions does not necessarily have to be the highest that is technically possible. With this test case, the optimal one, which yields the least iterations, is 8.

3.7.2 Test case n°2

For the test case hereafter, we consider $\Omega = [0,1]$, define the right hand side f as the constant function taking the value 1 in Ω and the Dirichlet boundary condition is prescribed on $\partial\Omega$ by the function:

$$\underline{u}_0(x,y) = \sqrt{x}; \tag{3.52}$$

whereas the diffusivity is defined as,

$$c^{(3)}(x,y) = \begin{cases} 10^5(\lfloor 9x \rfloor + 1) & \text{if} \quad \lfloor (9x) \rfloor \equiv 0 \pmod 2 \text{ and } \lfloor (9y) \rfloor \equiv 0 \pmod 2. \\ 1 & \text{otherwise.} \end{cases}$$

This second test case is similar to the third one of Chapter 2 where the error is scattered on the domain. By contrast, the mesh size and the size of the set L were increased. Tables 3.3 and 3.4 present the computational results of this test case.

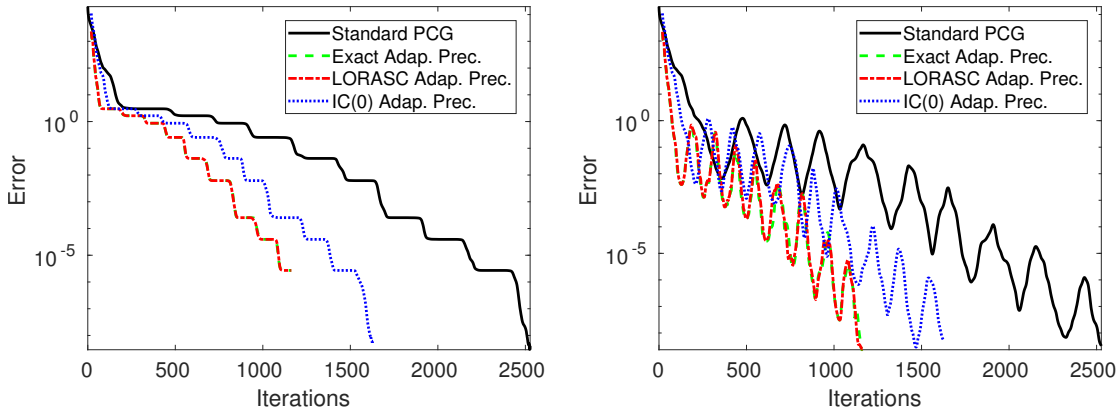
Table 3.3 – *Weak scaling results for test case n°2. The number of iterations are given for exact and approximate adaptive preconditioner using LORASC or Incomplete Cholesky IC(0).*

n	nnz	n_L	$\kappa(\mathbf{A}_L)$	N	n_Γ	$nnz(\mathbf{W}_\Gamma)$	m	n_{ev}	N_{mult}	it_{LORASC}	$it_{\text{IC}(0)}$	it_{exact}	it_{BJ}
14 690	100 650	10 281	6 008	8	183	12 321	24	4	58	929	1067	920	1569
41 699	288 725	26 584	62 546	16	1 024	262 228	136	32	262	1 143	1 607	1 142	2 527
78 428	544 350	50 000	8 790	32	1 633	170 853	216	23	403	2 227	2 456	2 181	3 541

Table 3.4 – *Strong scaling results for test case n°2.*

n	nnz	n_L	N	n_Γ	$nnz(\mathbf{W}_\Gamma)$	m	n_{ev}	N_{mult}	it_{LORASC}
78 428	544 350	50 000	4	96	4 626	20	2	44	2 214
78 428	544 350	50 000	8	290	29 244	38	5	81	2 185
78 428	544 350	50 000	16	687	72 003	90	12	171	2 186
78 428	544 350	50 000	32	1 633	170 853	216	23	403	2 227

Figure 3.2 illustrates how the error norms change over iterations for test case n°2 with standard solve procedure, adaptive solve procedure and variants of this latter. The matrix size and parameters' settings considered to plot the curves are indicated in the second row of Table 3.3.



(a) Energy norm of the algebraic error *(b) L-norm of the algebraic error*

Figure 3.2 – *Error evolution for test case n°1*

From the results shown in Tables 3.3 and 3.4 and Figure 3.2, we see that the behavior of the LORASC-based adaptive preconditioner is very close to that of the exact one, whereas the $IC(0)$-based one is slightly less efficient but still way better than the standard PCG with Block-Jacobi preconditioner.

It is also relevant to take a look at the number of matrix-vector operations needed for the Implicitly Restarted Lanczos Method in order to check that the computation of eigenvalues is not excessively costly. Indeed, the results reported in Tables 3.3, 3.4, 3.1, and 3.2 confirm this expectation, as N_{mult} is roughly double the dimension of the search subspace ($\approx 2m$) which is small with respect to n_L (or n_Γ).

3.8 Conclusion

We have proposed in this chapter the approximate adaptive preconditioners as an alternative to exact adaptive preconditioners. The latter have the advantage of efficiency, but the former entails lower computational costs. We have discussed two examples of approximate adaptive preconditioners, one based on Incomplete Cholesky with no fill-in, and the other one based on a LORASC preconditioner. The numerical results presented above clearly demonstrate that the efficiency of the approximate adaptive preconditioners depends on the type of approximation considered. In this sense, we can say that the LORASC-based preconditioner behaves better than the $IC(0)$-based one. In fact, it should be emphasized that the LORASC adaptive preconditioner yields an error reduction that is very close to the exact case by computing a few eigenvalues only. In sum, and in light of these observations, LORASC adaptive preconditioner appears to be a good candidate to replace the exact adaptive preconditioner as it has proven its ability to behave almost as robustly as the latter but with lower computation costs.

4

Adaptive a posteriori error estimates-based preconditioner for controlling a local algebraic error norm

Contents

Abstract

In this chapter, we introduce another variant for the adaptive preconditioner for iterative solution of sparse linear systems arising from partial differential equations with self-adjoint operators. With such a preconditioner, we prove that within a fixed-point iteration scheme, the growth rate of a dominant part of the algebraic error can be controlled. In fact, after deriving an error-based block partitioning of the matrix, where we denote by L the node indices where the algebraic error is large, we bound this dominant part of the error by a seminorm of the error and then demonstrate that the decrease of this latter quantity depends on the largest eigenvalue of the L-block of the preconditioned operator $\mathbf{M}^{-1}\mathbf{A}$. We present some numerical results to show that link, and lastly, we test this preconditioner with a PCG solver and compare against the first variant of the adaptive preconditioner.

Keywords— Adaptive preconditioner, fixed-point iteration, error's growth rate, block partitioning

4.1 Introduction

We recall the study context introduced previously. We are interested in solving a linear system $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ by an iterative method. The system stems from the discretization of a partial differential equation for which an error analysis has been undertaken and a posteriori algebraic error estimates are available. The magnitudes of these estimates indicate that the algebraic error is concentrated on some specific areas of the domain, which leads to a certain error-driven domain decomposition.

In Chapter 2, we have gathered the areas where the algebraic error was high in a subset of indices L. Then we have tried to find a way to efficiently reduce the energy norm of the error by expressing it as a sum of two terms : a first term that depends on L, that we call L-term and which is dominant –according to the information stemming from a posteriori error estimates– and a second term that does not depend on L that we call R-term (see (2.25)). The first term is presented in two forms. On one hand, it can be expressed as a scalar product of the projections of the residual and the error on the nodes of L. On the other hand, it can also be split into the sum of a norm of the error projected on the indices of L, and a coupling term (see (2.26)). As this latter term is also involved in the formula of the R-term, it is quite natural to seek to lessen the impact of the norm of the error projected on the indices of L. Yet, we were not able to devise a strategy to reduce it as the error vector remains generally unknown. That brought us to slightly change the target criterion and focus on the projection of the residual on the nodes of L. Making this projection nil implies that the L-term is cancelled (equal to zero). For this purpose, we have introduced an adequate adaptive solve procedure and its equivalent preconditioner and initial guess for PCG solver. We have explained that for this solver and with appropriate preconditioner and initial guess, it was possible to cancel a projection of the residual, but limiting a projection of the error was a complicated matter as the search directions in conjugate gradient methods are chosen to reduce the global energy norm of the error, not a part of it.

This chapter is organized as follows: Section 4.2 recalls the starting assumption on the algebraic error, then we discuss in Section 4.3 a first approach to locally reduce the localized large algebraic errors on the basis of the orthogonality properties of the preconditioned conjugate gradient (PCG) solver. In Section 4.4, we analyze the behavior of the partial error within a fixed-point iteration scheme. In Section 4.5, we derive specific preconditioners that would ensure that from an iteration

i to an iteration $i+1$, the evolution of that algebraic error localized on the targeted subdomains is controlled. More precisely, the growth rate of the local algebraic error between those two iterations can be bounded by a fixed coefficient. Section 4.6 makes the connection with the preconditioner derived in Chapter 2. The behaviors of both preconditioners are examined through some numerical tests presented in Section 4.7.

4.2 Preliminaries

We consider the same model problem dealt with in Section 2.2 of Chapter 2 and reuse the relevant notation in what follows. We also take up the starting assumption of a domain decomposition that is based on the algebraic error, i.e. we can estimate the local distribution of the error on all mesh elements and consequently decompose the main domain Ω into two disjoint subdomains Ω_1 and Ω_2 such that:

$$\begin{cases} \overline{\Omega}_1 \cup \overline{\Omega}_2 & = \overline{\Omega} \\ \Omega_1^o \cap \Omega_2^o & = \emptyset \end{cases} \tag{4.1}$$

where Ω_1 is the part with the high algebraic error:

$$\boxed{||\underline{\mathbf{K}}^{1/2}\nabla(\underline{u}_h - \underline{u}_h^{(i)})||^2_{\mathrm{L}^2(\Omega_1)} \gg ||\underline{\mathbf{K}}^{1/2}\nabla(\underline{u}_h - \underline{u}_h^{(i)})||^2_{\mathrm{L}^2(\Omega_2)}} \tag{4.2}$$

According to the domain decomposition of (4.1), we denote by $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$ the local stiffness matrices for the subdomains Ω_1 and Ω_2, respectively. While for the matrix $\mathbf{A}$ we have

$$\mathbf{A}_{jk} = (\underline{\mathbf{K}}^{\frac{1}{2}}\nabla\varphi_k, \underline{\mathbf{K}}^{\frac{1}{2}}\nabla\varphi_j), \qquad 1 \leq j,k \leq n\ ;$$

we define $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$ the Neumann matrices local to subdomains Ω_1 and Ω_2 respectively:

$$\mathbf{A}^{(1)}_{jk} = (\underline{\mathbf{K}}^{\frac{1}{2}}\nabla\varphi_k, \underline{\mathbf{K}}^{\frac{1}{2}}\nabla\varphi_j)_{\Omega_1},\ 1 \leq j,k \leq n,\ \text{supp}\,\varphi_k \cap \Omega_1 \neq \emptyset,\ \text{supp}\,\varphi_j \cap \Omega_1 \neq \emptyset$$

$$\mathbf{A}^{(2)}_{jk} = (\underline{\mathbf{K}}^{\frac{1}{2}}\nabla\varphi_k, \underline{\mathbf{K}}^{\frac{1}{2}}\nabla\varphi_j)_{\Omega_2},\ 1 \leq j,k \leq n,\ \text{supp}\,\varphi_k \cap \Omega_2 \neq \emptyset,\ \text{supp}\,\varphi_j \cap \Omega_2 \neq \emptyset$$

For the ease of presentation we assume a convenient ordering such that the variables corresponding to the vertices of Ω_1 are numbered first and those of Ω_2 second. Then we can split the original operator, represented algebraically by the stiffness matrix $\mathbf{A}$, as follows

$$\mathbf{A} = \mathbf{A}_p^{(1)} + \mathbf{A}_p^{(2)}, \tag{4.3}$$

where $\mathbf{A}_p^{(1)}, \mathbf{A}_p^{(2)}$ are symmetric positive semidefinite (SPSD) with the following shapes:

$$\mathbf{A}_p^{(1)} = \left(\begin{array}{c|c} \mathbf{A}^{(1)} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right); \qquad \mathbf{A}_p^{(2)} = \left(\begin{array}{c|c} \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{A}^{(2)} \end{array}\right).$$

They are the extensions of the local stiffness matrices (also called Neumann matrices [47]) $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$ to the whole domain, see Formula (2.21) and Figure 2.2 in Chapter 2. Indeed, if we define the

restriction matrices $\mathbf{R}_1$ and $\mathbf{R}_2$ from the global set of degrees of freedom to the set of degrees of freedom related to Ω_1 and to Ω_2 respectively, then

$$\begin{aligned} \mathbf{A}_p^{(1)} &= \mathbf{R}_1^{\mathrm{T}} \mathbf{A}^{(1)} \mathbf{R}_1; \\ \mathbf{A}_p^{(2)} &= \mathbf{R}_2^{\mathrm{T}} \mathbf{A}^{(2)} \mathbf{R}_2; \end{aligned} \tag{4.4}$$

Let $n_L \in \mathbb{N}$ be the number of nodes in subdomain Ω_1, then with a convenient node ordering, we have

$$\forall (\mathbf{x}_L, \mathbf{x}_R) \in \mathbb{R}^{n_L} \times \mathbb{R}^{n_R} : \quad \mathbf{R}_1 \cdot \begin{pmatrix} \mathbf{x}_L \\ \mathbf{x}_R \end{pmatrix} = \mathbf{x}_L. \tag{4.5}$$

Furthermore, we obtain the equivalent formulation to (4.2) in the realm of matrices:

$$\boxed{(\mathbf{x} - \mathbf{x}^{(i)})^{\mathrm{T}} \cdot \mathbf{A}_p^{(1)} \cdot (\mathbf{x} - \mathbf{x}^{(i)}) \gg (\mathbf{x} - \mathbf{x}^{(i)})^{\mathrm{T}} \cdot \mathbf{A}_p^{(2)} \cdot (\mathbf{x} - \mathbf{x}^{(i)})}, \tag{4.6}$$

where $\mathbf{x}^{(i)}$ is the approximate solution at iteration i. Note that summing both sides of Inequality (4.6) gives the energy norm of the error: $(\mathbf{x} - \mathbf{x}^{(i)})^{\mathrm{T}} \cdot \mathbf{A} \cdot (\mathbf{x} - \mathbf{x}^{(i)})$.

4.3 Local error reduction with PCG

Inequality (4.6) expresses that $\mathbf{A}_p^{(1)}$-seminorm of the error is the dominant part of the energy norm of the error. Thus, it is this quantity that should be targeted and decreased. For this purpose, we are going to discuss the orthogonality property that allows a decrease of the energy norm of the error in PCG solver, then we try to derive a similar orthogonality property to ensure the reduction of the dominant share of the algebraic error, which is localized in $\overline{\Omega}_1$.
In what follows, for Euclidean vectors $\mathbf{a}$ and $\mathbf{b}$ of size m $(\mathbf{a}, \mathbf{b})$ denotes a dot product, and for a symmetric matrix $\mathbf{B}$ of dimension $m \times m$, we write $(\mathbf{a}, \mathbf{b})_{\mathbf{B}} := (\mathbf{a}, \mathbf{B} \cdot \mathbf{b})$.
Let $\mathbf{x}^{(i)}$ be the approximate solution of (2.4) obtained at iteration i, with a PCG method using a preconditioner $\mathbf{M}$.

Definition 4.1. Let $\mathbf{B}$ be a SPSD matrix and $(\mathbf{x}^{(j)})_j$ a sequence of vectors, $\mathbf{x}^{(j)} \underset{j}{\longrightarrow} \mathbf{x}$. We say that a $\mathbf{B}$-orthogonality property is satisfied when for every iteration j, we have

$$(\mathbf{B} \cdot (\mathbf{x} - \mathbf{x}^{(j+1)}), \mathbf{x}^{(j)} - \mathbf{x}^{(j+1)}) = 0 \, . \tag{4.7}$$

Lemma 4.1. *Let $\mathbf{B}$ be a SPSD matrix. The $\mathbf{B}$-orthogonality property satisfied for a sequence of vectors $(\mathbf{x}^{(j)})_j$ approximating the vector $\mathbf{x}$, ensures the decrease of the $\mathbf{B}$-seminorm of the algebraic error $(\mathbf{B} \cdot (\mathbf{x} - \mathbf{x}^{(j)}), \mathbf{x} - \mathbf{x}^{(j)})$ from any iteration j to iteration $j+1$.*

Proof. In fact, due to $\mathbf{B}$-orthogonality we have for every iteration j:

$$(\mathbf{x} - \mathbf{x}^{(j+1)}, \mathbf{x}^{(j)} - \mathbf{x}^{(j+1)})_{\mathbf{B}} = 0$$

And

$$\begin{aligned}||\mathbf{x}-\mathbf{x}^{(j)}||_{\mathbf{B}}^2 &= ||\mathbf{x}-\mathbf{x}^{(j+1)}||_{\mathbf{B}}^2 + ||\mathbf{x}^{(j)}-\mathbf{x}^{(j+1)}||_{\mathbf{B}}^2 - 2*(\mathbf{x}-\mathbf{x}^{(j+1)},\mathbf{x}^{(j)}-\mathbf{x}^{(j+1)})_{\mathbf{B}} \\ &= ||\mathbf{x}-\mathbf{x}^{(j+1)}||_{\mathbf{B}}^2 + ||\mathbf{x}^{(j)}-\mathbf{x}^{(j+1)}||_{\mathbf{B}}^2 \\ &\geq ||\mathbf{x}-\mathbf{x}^{(j+1)}||_{\mathbf{B}}^2\end{aligned}$$

□

While iteratively solving (2.4), we could seek two orthogonalities in particular for the following reasons:

- **A-orthogonality**: It allows to minimize the global energy norm of the error according to Lemma 4.1.
- **$\mathbf{A}_p^{(1)}$-orthogonality**: It allows, according to Lemma 4.1, to reduce the dominant part of the global energy norm of the error as assumed in (2.19).

In our context, a primary goal is to reduce the $\mathbf{A}_p^{(1)}$-seminorm of the algebraic error, which is dominant according to the starting assumption (2.19). As stated in Lemma 4.1, the $\mathbf{A}_p^{(1)}$-orthogonality is a sufficient condition for the decrease of those quantities. For this reason, we now investigate means of ensuring the $\mathbf{A}_p^{(1)}$-orthogonality property. The $\mathbf{A}$-orthogonality is satisfied by definition thanks to the properties of the PCG method. Since we prefer to stay within the framework of PCG, there is no room in the choice of search directions. But the step size configuration constitutes a point for reflection, in the sense that there could exist some preconditioner $\mathbf{M}$ which yields particular step sizes such that the $\mathbf{A}_p^{(1)}$-orthogonality holds too.

Lemma 4.2. *Consider a PCG iterative process [123, Algorithm 9.1,Chapter 9] to solve the linear system (2.4), and denote by $\mathbf{r}^{(j)}$ and $\mathbf{p}^{(j)}$ the residual and descent direction respectively at iteration j. The $\mathbf{A}$-orthogonality property is guaranteed by the step size α_j:*

$$\alpha_j = \frac{{\mathbf{r}^{(j)}}^{\mathrm{T}} \cdot \mathbf{M}^{-1} \cdot \mathbf{r}^{(j)}}{{\mathbf{p}^{(j)}}^{\mathrm{T}} \cdot \mathbf{A} \cdot \mathbf{p}^{(j)}} = \frac{{\mathbf{p}^{(j)}}^{\mathrm{T}} \cdot \mathbf{r}^{(j)}}{{\mathbf{p}^{(j)}}^{\mathrm{T}} \cdot \mathbf{A} \cdot \mathbf{p}^{(j)}} \; ; \tag{4.8}$$

whereas the $\mathbf{A}_p^{(1)}$-orthogonality property can be guaranteed by the value taken by the step size α_j:

$$\alpha_j = \frac{{\mathbf{p}^{(j)}}^{\mathrm{T}} \cdot \mathbf{A}_p^{(1)} \cdot (\mathbf{x}-\mathbf{x}^{(j)})}{{\mathbf{p}^{(j)}}^{\mathrm{T}} \cdot \mathbf{A}_p^{(1)} \cdot \mathbf{p}^{(j)}} \; . \tag{4.9}$$

Proof. The recurrence formulas of PCG give:

$$\begin{cases} \mathbf{x}^{(j+1)} = \mathbf{x}^{(j)} + \alpha_j \mathbf{p}^{(j)} \\ \mathbf{r}^{(j+1)} = \mathbf{r}^{(j)} - \alpha_j \mathbf{A} \cdot \mathbf{p}^{(j)} \end{cases}$$

Therefore,

$$\begin{aligned}(\mathbf{x}^{(j)}-\mathbf{x}^{(j+1)})^{\mathrm{T}}\cdot\mathbf{A}\cdot(\mathbf{x}-\mathbf{x}^{(j+1)})=0 &\iff (\alpha_j\mathbf{p}^{(j)})^{\mathrm{T}}\cdot\mathbf{r}^{(j+1)}=0\\ &\iff {\mathbf{p}^{(j)}}^{\mathrm{T}}\cdot(\mathbf{r}^{(j)}-\alpha_j\mathbf{A}\cdot\mathbf{p}^{(j)})=0\\ &\iff \alpha_j=\frac{{\mathbf{p}^{(j)}}^{\mathrm{T}}\cdot\mathbf{r}^{(i)}}{{\mathbf{p}^{(j)}}^{\mathrm{T}}\cdot\mathbf{A}\cdot\mathbf{p}^{(j)}}\end{aligned}$$

We can prove the $\mathbf{A}_p^{(1)}$-orthogonality with (4.9) analogously. The right part of the equality (4.8) can be demonstrated by induction from the recurrence formulas of PCG expressed above. □

Lemma 4.2 gives step sizes that ensure $\mathbf{A}$-orthogonality and $\mathbf{A}_p^{(1)}$-orthogonality, respectively. Naturally, one can wonder when the two expressions (4.8) and (4.9) above match, i.e.:

$$\frac{{\mathbf{p}^{(j)}}^{\mathrm{T}}\cdot\mathbf{r}^{(j)}}{{\mathbf{p}^{(j)}}^{\mathrm{T}}\cdot\mathbf{A}\cdot\mathbf{p}^{(j)}}=\frac{{\mathbf{p}^{(j)}}^{\mathrm{T}}\cdot\mathbf{A}_p^{(1)}\cdot(\mathbf{x}-\mathbf{x}^{(j)})}{{\mathbf{p}^{(j)}}^{\mathrm{T}}\cdot\mathbf{A}_p^{(1)}\cdot\mathbf{p}^{(j)}} \tag{4.11}$$

The formula above cannot be used in practice because the term $\mathbf{A}_p^{(1)}\cdot(\mathbf{x}-\mathbf{x}^{(j)})$ cannot be computed as $\mathbf{x}$ is unknown. We outline here a special case when (4.11) holds: when $\mathbf{x}-\mathbf{x}^{(0)}$ is an eigenvector of $\mathbf{M}^{-1}\mathbf{A}$. Indeed, let $\lambda\in\mathbb{R}^*$ be the associated eigenvalue. We have:

$$\mathbf{p}^{(0)}:=\mathbf{M}^{-1}\cdot\mathbf{r}^{(0)}=\mathbf{M}^{-1}\mathbf{A}\cdot(\mathbf{x}-\mathbf{x}^{(0)})=\lambda(\mathbf{x}-\mathbf{x}^{(0)})\quad\text{and}\quad\mathbf{A}\cdot\mathbf{p}^{(0)}=\lambda\mathbf{r}^{(0)}$$

Hence,

$$\frac{{\mathbf{p}^{(0)}}^{\mathrm{T}}\cdot\mathbf{r}^{(0)}}{{\mathbf{p}^{(0)}}^{\mathrm{T}}\cdot\mathbf{A}\cdot\mathbf{p}^{(0)}}=\frac{1}{\lambda}\quad\text{and}\quad\frac{{\mathbf{p}^{(0)}}^{\mathrm{T}}\cdot\mathbf{A}_p^{(0)}\cdot(\mathbf{x}-\mathbf{x}^{(0)})}{{\mathbf{p}^{(0)}}^{\mathrm{T}}\cdot\mathbf{A}_p^{(0)}\cdot\mathbf{p}^{(0)}}=\frac{1}{\lambda}\,.$$

However, this assumption is too strong to be satisfied in practice for a PCG solver. In fact, it allows for the convergence in one iteration because:

$$\mathbf{M}^{-1}\mathbf{A}\cdot(\mathbf{x}-\mathbf{x}^{(0)})=\lambda(\mathbf{x}-\mathbf{x}^{(0)})\Longrightarrow\begin{cases}\mathbf{z}^{(0)}:=\mathbf{M}^{-1}\cdot\mathbf{r}^{(0)}=\lambda(\mathbf{x}-\mathbf{x}^{(0)});\\ \mathbf{p}^{(0)}:=\mathbf{z}^{(0)}=\lambda(\mathbf{x}-\mathbf{x}^{(0)});\\ \alpha^{(0)}:=\dfrac{{\mathbf{r}^{(0)}}^{\mathrm{T}}\cdot\mathbf{z}^{(0)}}{{\mathbf{p}^{(0)}}^{\mathrm{T}}\cdot\mathbf{A}\cdot\mathbf{p}^{(0)}}\\ \quad=\dfrac{\lambda}{\lambda^2}\times\dfrac{(\mathbf{x}-\mathbf{x}^{(0)})^{\mathrm{T}}\mathbf{A}(\mathbf{x}-\mathbf{x}^{(0)})}{(\mathbf{x}-\mathbf{x}^{(0)})^{\mathrm{T}}\mathbf{A}(\mathbf{x}-\mathbf{x}^{(0)})}=\dfrac{1}{\lambda}\end{cases}$$
$$\Longrightarrow\mathbf{x}^{(1)}:=\mathbf{x}^{(0)}+\alpha^{(0)}\mathbf{p}^{(0)}=\mathbf{x}$$

This reflects the difficulty of reducing the $\mathbf{A}_p^{(1)}$ seminorm of the error with a PCG solver, and motivates seeking another procedure to ensure the local reduction of dominant errors.

4.4 Controlling the local algebraic error in fixed-point iteration scheme

Controlling the evolution of the left hand side of (4.6), which is the dominant part of the energy norm of the error, from an iteration to another is limited and kept under a given threshold is equivalent to ensuring the following property:

$$\exists \tau > 0, \quad \forall \mathbf{i} \in \mathbb{N}, \quad (\mathbf{x} - \mathbf{x}^{(i+1)})^{\mathrm{T}} \cdot \mathbf{A}_p^{(1)} \cdot (\mathbf{x} - \mathbf{x}^{(i+1)}) \leq \tau (\mathbf{x} - \mathbf{x}^{(i)})^{\mathrm{T}} \cdot \mathbf{A}_p^{(1)} \cdot (\mathbf{x} - \mathbf{x}^{(i)}).$$

Thus considering a fixed-point iteration scheme

$$\mathbf{x}^{(i+1)} := \mathbf{x}^{(i)} + \mathbf{M}^{-1} \cdot (\mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(i)}), \forall i \in \mathbb{N}; \tag{4.12}$$

with an arbitrary initial guess $\mathbf{x}^{(0)}$ brings us to search for a preconditioner $\mathbf{M}^{-1}$ that satisfies the property:

$$\exists \tau > 0, \quad \forall \mathbf{u} \in \mathbb{R}^n : \quad ((\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}) \cdot \mathbf{u})^{\mathrm{T}} \cdot \mathbf{A}_p^{(1)} \cdot ((\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}) \cdot \mathbf{u}) \leq \tau \mathbf{u}^{\mathrm{T}} \cdot \mathbf{A}_p^{(1)} \cdot \mathbf{u}; \tag{4.13}$$

because due to (4.12), we have $\mathbf{x} - \mathbf{x}^{(i+1)} = (\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}) \cdot (\mathbf{x} - \mathbf{x}^{(i)})$.

In what follows, we state some lemmas that are useful for establishing the necessary and sufficient conditions for property (4.13).

Lemma 4.3. *Let* $\mathbf{P}$ *be a symmetric positive semi-definite matrix,* $\mathbf{Q}$ *be an invertible matrix of size* $n \in \mathbb{N}$ *and* V *be an arbitrary subspace of* $\mathbb{R}^n$*. The following two assertions are equivalent:*

— $\exists \tau_1 > 0, \quad \forall \mathbf{u} \in V : \quad ((\mathbf{I} - \mathbf{Q}) \cdot \mathbf{u})^{\mathrm{T}} \cdot \mathbf{P} \cdot ((\mathbf{I} - \mathbf{Q}) \cdot \mathbf{u}) \leq \tau_1 \mathbf{u}^{\mathrm{T}} \cdot \mathbf{P} \cdot \mathbf{u};$

— $\exists \tau_2 > 0, \quad \forall \mathbf{u} \in V : \quad (\mathbf{Q} \cdot \mathbf{u})^{\mathrm{T}} \cdot \mathbf{P} \cdot (\mathbf{Q} \cdot \mathbf{u}) \leq \tau_2 \mathbf{u}^{\mathrm{T}} \cdot \mathbf{P} \cdot \mathbf{u};$

Proof. We denote by $||\ ||_{\mathbf{P}} : \mathbf{x} \mapsto \sqrt{\mathbf{x}^{\mathrm{T}} \cdot \mathbf{P} \cdot \mathbf{x}}$ the seminorm defined by $\mathbf{P}$ on $\mathbb{R}^n$. To prove the equivalence of the assertions, it suffices to notice that for any $\mathbf{u} \in \mathbb{R}^n$ we have:

$$||\mathbf{u} - \mathbf{Q} \cdot \mathbf{u}||_{\mathbf{P}}^2 \leq 2||\mathbf{u}||_{\mathbf{P}}^2 + 2||\mathbf{Q} \cdot \mathbf{u}||_{\mathbf{P}}^2$$

and

$$||\mathbf{Q} \cdot \mathbf{u}||_{\mathbf{P}}^2 = ||\mathbf{u} - (\mathbf{u} - \mathbf{Q} \cdot \mathbf{u})||_{\mathbf{P}}^2 \leq 2||\mathbf{u}||_{\mathbf{P}}^2 + 2||\mathbf{u} - \mathbf{Q} \cdot \mathbf{u}||_{\mathbf{P}}^2.$$

□

Lemma 4.4. *Let* $\mathbf{P}$ *be a symmetric positive semi-definite matrix and* $\mathbf{Q}$ *be an invertible matrix of size* $n \in \mathbb{N}$*. If*

$$\exists \tau > 0, \quad \forall \mathbf{u} \in \mathbb{R}^n : \quad ((\mathbf{I} - \mathbf{Q}) \cdot \mathbf{u})^{\mathrm{T}} \cdot \mathbf{P} \cdot ((\mathbf{I} - \mathbf{Q}) \cdot \mathbf{u}) \leq \tau \mathbf{u}^{\mathrm{T}} \cdot \mathbf{P} \cdot \mathbf{u};$$

then $\mathrm{Ker}(\mathbf{P})$ *is invariant of* $\mathbf{Q}$*, i.e.* $\quad \forall \mathbf{v} \in \mathrm{Ker}(\mathbf{P}) : \mathbf{Q} \cdot \mathbf{v} \in \mathrm{Ker}(\mathbf{P})$.

Proof. We give a proof by contrapositive of this lemma. If $\mathrm{Ker}(\mathbf{P})$ is not invariant of $\mathbf{Q}$ then there

exists a vector $\mathbf{u}_0 \in \mathrm{Ker}(\mathbf{P})$ such that $\mathbf{Q}\cdot\mathbf{u}_0 \notin \mathrm{Ker}(\mathbf{P})$. Thus:

$$\forall \tau > 0, \quad \exists \mathbf{u}_0 \in \mathbb{R}^n : \quad ((\mathbf{I}-\mathbf{Q})\cdot\mathbf{u}_0)^{\mathrm{T}}\cdot\mathbf{P}\cdot((\mathbf{I}-\mathbf{Q})\cdot\mathbf{u}_0) = ||\mathbf{Q}\cdot\mathbf{u}_0||^2_{\mathbf{P}} > 0 = \tau(\mathbf{u}_0^{\mathrm{T}}\cdot\mathbf{P}\cdot\mathbf{u}_0).$$

□

The next corollary follows from the above lemma by taking $\mathbf{P} := \mathbf{A}_p^{(1)}$ and $\mathbf{Q} := \mathbf{M}^{-1}\mathbf{A}$.

Corollary 4.1. *Let* $\mathbf{M}^{-1}$ *be a preconditioner of the matrix* $\mathbf{A}$ *that satisfies:*

$$\exists \tau > 0, \quad \forall \mathbf{u} \in \mathbb{R}^n : \quad ((\mathbf{I}-\mathbf{M}^{-1}\mathbf{A})\cdot\mathbf{u})^{\mathrm{T}}\cdot\mathbf{A}_p^{(1)}\cdot((\mathbf{I}-\mathbf{M}^{-1}\mathbf{A})\cdot\mathbf{u}) \leq \tau\mathbf{u}^{\mathrm{T}}\cdot\mathbf{A}_p^{(1)}\cdot\mathbf{u};$$

then $\mathrm{Ker}(\mathbf{A}_p^{(1)})$ *is invariant of* $\mathbf{M}^{-1}\mathbf{A}$.

This corollary provides a necessary condition for the property (4.13) to be satisfied. On the other hand, we aim at proving sufficient condition for that property with the following lemma:

Lemma 4.5. *Let* $\mathbf{M}^{-1}$ *be a preconditioner of the matrix* $\mathbf{A}$.
If $\mathrm{Range}(\mathbf{A}_p^{(1)})$ *is invariant of* $(\mathbf{M}^{-1}\mathbf{A})^{\mathrm{T}}\mathbf{A}_p^{(1)}(\mathbf{M}^{-1}\mathbf{A})$ *then:*

$$\exists \tau > 0, \quad \forall \mathbf{u} \in \mathrm{Range}(\mathbf{A}_p^{(1)}) : \quad ((\mathbf{I}-\mathbf{M}^{-1}\mathbf{A})\cdot\mathbf{u})^{\mathrm{T}}\cdot\mathbf{A}_p^{(1)}\cdot((\mathbf{I}-\mathbf{M}^{-1}\mathbf{A})\cdot\mathbf{u}) \leq \tau\mathbf{u}^{\mathrm{T}}\cdot\mathbf{A}_p^{(1)}\cdot\mathbf{u};$$

Proof. $\mathbf{A}_p^{(1)}$ and $(\mathbf{M}^{-1}\mathbf{A})^{\mathrm{T}}\mathbf{A}_p^{(1)}(\mathbf{M}^{-1}\mathbf{A})$ are symmetric positive semidefinite.
In addition, if $\mathrm{Range}(\mathbf{A}_p^{(1)})$ is invariant of $(\mathbf{M}^{-1}\mathbf{A})^{\mathrm{T}}\mathbf{A}_p^{(1)}(\mathbf{M}^{-1}\mathbf{A})$ then, $(\mathbf{M}^{-1}\mathbf{A})^{\mathrm{T}}\mathbf{A}_p^{(1)}(\mathbf{M}^{-1}\mathbf{A})$ and $\mathbf{A}_p^{(1)}$, seen as linear operators from $\mathrm{Range}(\mathbf{A}_p^{(1)})$ to $\mathrm{Range}(\mathbf{A}_p^{(1)})$, are symmetric positive semidefinite and symmetric positive definite respectively. In this case, we can introduce the following generalized eigenvalue problem:

$$\mathrm{Find}(\mathbf{y}_k, \mu_k) \in \mathrm{Range}(\mathbf{A}_p^{(1)}) \times \mathbb{R} \quad \text{such that} \quad (\mathbf{M}^{-1}\mathbf{A})^{\mathrm{T}}\mathbf{A}_p^{(1)}(\mathbf{M}^{-1}\mathbf{A})\cdot\mathbf{y}_k = \mu_k\mathbf{A}_p^{(1)}\cdot\mathbf{y}_k. \quad (4.14)$$

Since $\mathbf{A}_p^{(1)}$ is a SPD operator on $\mathrm{Range}(\mathbf{A}_p^{(1)})$, the eigenvalues of (4.14) can be chosen so that they form a basis of $\mathrm{Range}(\mathbf{A}_p^{(1)})$ that is both $\mathbf{A}_p^{(1)}$-orthonormal and $(\mathbf{M}^{-1}\mathbf{A})^{\mathrm{T}}\mathbf{A}_p^{(1)}(\mathbf{M}^{-1}\mathbf{A})$-orthogonal. Let $m = \dim(\mathrm{Range}(\mathbf{A}_p^{(1)})) = \mathrm{rank}(\mathbf{A}_p^{(1)})$ and let $\mathbf{u} \in \mathrm{Range}(\mathbf{A}_p^{(1)})$ then:

$$\begin{aligned}
\mathbf{u} &= \sum_{k=1}^{m}(\mathbf{A}_p^{(1)}\cdot\mathbf{u}, \mathbf{y}_k)\mathbf{y}_k \\
(\mathbf{M}^{-1}\mathbf{A})^{\mathrm{T}}\mathbf{A}_p^{(1)}(\mathbf{M}^{-1}\mathbf{A})\cdot\mathbf{u} &= \sum_{k=1}^{m}(\mathbf{A}_p^{(1)}\cdot\mathbf{u}, \mathbf{y}_k)(\mathbf{M}^{-1}\mathbf{A})^{\mathrm{T}}\mathbf{A}_p^{(1)}(\mathbf{M}^{-1}\mathbf{A})\cdot\mathbf{y}_k = \sum_{k=1}^{m}\mu_k(\mathbf{A}_p^{(1)}\cdot\mathbf{u}, \mathbf{y}_k)\mathbf{A}_p^{(1)}\cdot\mathbf{y}_k \\
\mathbf{u}^{\mathrm{T}}\cdot(\mathbf{M}^{-1}\mathbf{A})^{\mathrm{T}}\mathbf{A}_p^{(1)}(\mathbf{M}^{-1}\mathbf{A})\cdot\mathbf{u} &= \sum_{k=1}^{m}\mu_k(\mathbf{A}_p^{(1)}\cdot\mathbf{u}, \mathbf{y}_k)^2 \\
&\leq (\max_{1\leq i\leq m}\mu_i)^2\sum_{k=1}^{m}(\mathbf{A}_p^{(1)}\cdot\mathbf{u}, \mathbf{y}_k)^2 = (\max_{1\leq i\leq m}\mu_i)^2\mathbf{u}^{\mathrm{T}}\cdot\mathbf{A}_p^{(1)}\cdot\mathbf{u}
\end{aligned}$$

And the result of Lemma 4.3 ends the proof. □

It must be emphasised that Lemma 4.5 gives a sufficient condition for Property (4.13) to be true not on the whole space $\mathbb{R}^n$ but only on a subspace of it. However, the error $\mathbf{e}^{(i)} := \mathbf{x} - \mathbf{x}^{(i)}$ is not guaranteed to belong to that subspace as the initial guess $\mathbf{x}^{(0)}$ is arbitrarily chosen. Therefore, this sufficient condition is too restrictive, since its effect is valid only for the iterations when the error lies in the range of $\mathbf{A}_p^{(1)}$. It is also worth mentioning that the necessary condition of Corollary 4.1 and the sufficient condition of Lemma 4.5 do not match. In the sequel, we will derive a second property that is similar to (4.13), involves a definite matrix and for which we can derive a sufficient and necessary condition.

4.5 Deriving a block partitioning and controlling the corresponding algebraic error norm

By proceeding in the same way as in Chapter 2, Section 2.4.2, we replace the sum-splitting of the operator, as in (4.3), by a block-partitioning of the matrix, such as

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_L & \mathbf{A}_{LR} \\ \mathbf{A}_{RL} & \mathbf{A}_R \end{pmatrix}, \tag{4.15}$$

where the L-part stands for the set of nodes that belong to Ω_1 and the R-part for the rest. n_L and n_R designate the sizes of the diagonal SPD blocks $\mathbf{A}_L$ and $\mathbf{A}_R$ respectively. We recall that in this case, if we denote:

$$\underline{\mathbf{A}}_L := \mathbf{R}_1^{\mathrm{T}} \mathbf{A}_L \mathbf{R}_1 = \begin{pmatrix} \mathbf{A}_L & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \tag{4.16}$$

then by combining the formulation of (4.6) with the superiority property of Lemma 2.1 in Chapter 2, which states that the $\mathbf{A}_L$-norm is higher than the $\mathbf{A}^{(1)}$-seminorm, we have:

$$(\mathbf{x} - \mathbf{x}^{(i)})^{\mathrm{T}} \cdot \underline{\mathbf{A}}_L \cdot (\mathbf{x} - \mathbf{x}^{(i)}) \geq (\mathbf{x} - \mathbf{x}^{(i)})^{\mathrm{T}} \cdot \mathbf{A}_p^{(1)} \cdot (\mathbf{x} - \mathbf{x}^{(i)}) \gg (\mathbf{x} - \mathbf{x}^{(i)})^{\mathrm{T}} \cdot \mathbf{A}_p^{(2)} \cdot (\mathbf{x} - \mathbf{x}^{(i)}) \tag{4.17}$$

Therefore, instead of controlling the dominant $\mathbf{A}_p^{(1)}$-seminorm of the error, we will focus on the $\underline{\mathbf{A}}_L$-seminorm of the error which is larger. To ensure that the evolution of this latter in a fixed-point iteration scheme from an iteration to another is limited and kept below a fixed coefficient, we have to find a preconditioner $\mathbf{M}^{-1}$ such that

$$\exists \tau > 0, \quad \forall \mathbf{u} \in \mathbb{R}^n : \quad ((\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}) \cdot \mathbf{u})^{\mathrm{T}} \cdot \underline{\mathbf{A}}_L \cdot ((\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}) \cdot \mathbf{u}) \leq \tau \mathbf{u}^{\mathrm{T}} \cdot \underline{\mathbf{A}}_L \cdot \mathbf{u}. \tag{4.18}$$

Lemma 4.4 applied to the matrices $\mathbf{P} := \underline{\mathbf{A}}_L$ and $\mathbf{Q} := \mathbf{M}^{-1}\mathbf{A}$ yields the following corollary.

Corollary 4.2. *Let* $\mathbf{M}^{-1}$ *be a preconditioner of the matrix* $\mathbf{A}$ *that satisfies*

$$\exists \tau > 0, \quad \forall \mathbf{u} \in \mathbb{R}^n : \quad ((\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}) \cdot \mathbf{u})^{\mathrm{T}} \cdot \underline{\mathbf{A}}_L \cdot ((\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}) \cdot \mathbf{u}) \leq \tau \mathbf{u}^{\mathrm{T}} \cdot \underline{\mathbf{A}}_L \cdot \mathbf{u},$$

then $\mathrm{Ker}(\underline{\mathbf{A}}_L)$ *is invariant of* $\mathbf{M}^{-1}\mathbf{A}$.

Lemma 4.6. $\mathrm{Ker}(\underline{\mathbf{A}}_L)$ *is invariant of* $\mathbf{M}^{-1}\mathbf{A}$ *if and only if*

$$\exists \mathbf{M}_1 \in \mathbb{R}^{n_L \times n_L}, \exists \mathbf{M}_3 \in \mathbb{R}^{n_R \times n_L}, \exists \mathbf{M}_4 \in \mathbb{R}^{n_R \times n_R} : \mathbf{M}^{-1}\mathbf{A} = \begin{pmatrix} \mathbf{M}_1 & \mathbf{0} \\ \mathbf{M}_3 & \mathbf{M}_4 \end{pmatrix}.$$

Proof. As $\underline{\mathbf{A}}_L = \begin{pmatrix} \mathbf{A}_L & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$ and $\mathbf{A}_L$ is SPD, we have: $\mathrm{Ker}(\underline{\mathbf{A}}_L) = \left\{ \begin{pmatrix} \mathbf{0} \\ \mathbf{y}_R \end{pmatrix} | \mathbf{y}_R \in \mathbb{R}^{n_R} \right\}$. Let $\mathbf{M}^{-1}\mathbf{A} = \begin{pmatrix} \mathbf{M}_1 & \mathbf{M}_2 \\ \mathbf{M}_3 & \mathbf{M}_4 \end{pmatrix}$, then $\underline{\mathbf{A}}_L\mathbf{M}^{-1}\mathbf{A} = \begin{pmatrix} \mathbf{A}_L\mathbf{M}_1 & \mathbf{A}_L\mathbf{M}_2 \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$. Let $\mathbf{y}_R \in \mathbb{R}^{n_R}$, we have:

$$\underline{\mathbf{A}}_L\mathbf{M}^{-1}\mathbf{A} \cdot \begin{pmatrix} \mathbf{0} \\ \mathbf{y}_R \end{pmatrix} = \begin{pmatrix} \mathbf{A}_L\mathbf{M}_2\mathbf{y}_R \\ \mathbf{0} \end{pmatrix}.$$

Therefore, $\mathrm{Ker}(\underline{\mathbf{A}}_L)$ is invariant of $\mathbf{M}^{-1}\mathbf{A}$ if and only if $\quad \forall \mathbf{y} \in \mathrm{Ker}(\underline{\mathbf{A}}_L), \quad \underline{\mathbf{A}}_L\mathbf{M}^{-1}\mathbf{A} \cdot \mathbf{y} = \mathbf{0}, \quad$ i.e.

$$\forall \mathbf{y}_R \in \mathbb{R}^{n_R}, \quad \mathbf{A}_L\mathbf{M}_2 \cdot \mathbf{y}_R = \mathbf{0}$$

Since $\mathbf{A}_L$ is SPD, this latter property is equivalent to:

$$\forall \mathbf{y}_R \in \mathbb{R}^{n_R}, \quad \mathbf{M}_2 \cdot \mathbf{y}_R = \mathbf{0}$$

which means that $\mathbf{M}_2 = \mathbf{0}$. □

Lemma 4.7. *Let* $\mathbf{M}^{-1}$ *be a preconditioner of the matrix* $\mathbf{A}$ *such that*

$$\mathbf{M}^{-1}\mathbf{A} = \begin{pmatrix} \mathbf{M}_1 & \mathbf{0} \\ \mathbf{M}_3 & \mathbf{M}_4 \end{pmatrix}, \tag{4.19}$$

where $\mathbf{M}_1 \in \mathbb{R}^{n_L \times n_L}, \mathbf{M}_3 \in \mathbb{R}^{n_R \times n_L}, \mathbf{M}_4 \in \mathbb{R}^{n_R \times n_R}$. *Then*

$$\exists \tau > 0, \quad \forall \mathbf{u} \in \mathbb{R}^n : \quad (\mathbf{M}^{-1}\mathbf{A} \cdot \mathbf{u})^{\mathrm{T}} \cdot \underline{\mathbf{A}}_L \cdot (\mathbf{M}^{-1}\mathbf{A} \cdot \mathbf{u}) \leq \tau \mathbf{u}^{\mathrm{T}} \cdot \underline{\mathbf{A}}_L \cdot \mathbf{u}.$$

Proof. With (4.19), we have

$$(\mathbf{M}^{-1}\mathbf{A})^{\mathrm{T}}\underline{\mathbf{A}}_L(\mathbf{M}^{-1}\mathbf{A}) = \begin{pmatrix} \mathbf{M}_1^{\mathrm{T}} & \mathbf{M}_3^{\mathrm{T}} \\ \mathbf{0} & \mathbf{M}_4^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} \mathbf{A}_L & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{M}_1 & \mathbf{0} \\ \mathbf{M}_3 & \mathbf{M}_4 \end{pmatrix} = \begin{pmatrix} \mathbf{M}_1^{\mathrm{T}}\mathbf{A}_L\mathbf{M}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}.$$

Let $\mathbf{u} = \begin{pmatrix} \mathbf{u}_L \\ \mathbf{u}_R \end{pmatrix} \in \mathbb{R}^n$, we have:

$$\mathbf{u}^{\mathrm{T}} \cdot (\mathbf{M}^{-1}\mathbf{A})^{\mathrm{T}}\underline{\mathbf{A}}_L(\mathbf{M}^{-1}\mathbf{A}) \cdot \mathbf{u} = \begin{pmatrix} \mathbf{u}_L \\ \mathbf{u}_R \end{pmatrix}^{\mathrm{T}} \cdot \begin{pmatrix} \mathbf{M}_1^{\mathrm{T}}\mathbf{A}_L\mathbf{M}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{u}_L \\ \mathbf{u}_R \end{pmatrix} = \mathbf{u}_L^{\mathrm{T}} \cdot \mathbf{M}_1^{\mathrm{T}}\mathbf{A}_L\mathbf{M}_1 \cdot \mathbf{u}_L$$

and

$$\mathbf{u}^{\mathrm{T}} \cdot \underline{\mathbf{A}}_L \cdot \mathbf{u} = \begin{pmatrix} \mathbf{u}_L \\ \mathbf{u}_R \end{pmatrix}^{\mathrm{T}} \cdot \begin{pmatrix} \mathbf{A}_L & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{u}_L \\ \mathbf{u}_R \end{pmatrix} = \mathbf{u}_L^{\mathrm{T}} \cdot \mathbf{A}_L \cdot \mathbf{u}_L.$$

We consider the following generalized eigenvalue problem

$$\text{find} \quad (\lambda_k, \mathbf{y}_k) \in \mathbb{R} \times \mathbb{R}^{n_L} \quad \text{such that} \quad \mathbf{M}_1^{\mathrm{T}} \mathbf{A}_L \mathbf{M}_1 \cdot \mathbf{y}_k = \lambda_k \mathbf{A}_L \cdot \mathbf{y}_k. \tag{4.20}$$

$\mathbf{M}_1^{\mathrm{T}} \mathbf{A}_L \mathbf{M}_1$ and $\mathbf{A}_L$ are both symmetric positive definite matrices, therefore the eigenvectors $\mathbf{y}_k$ can be chosen so that they form a basis of $\mathbb{R}^{n_L}$ that is both $\mathbf{A}_L$-orthonormal and $\mathbf{M}_1^{\mathrm{T}} \mathbf{A}_L \mathbf{M}_1$-orthogonal. As a consequence, we can write:

$$\begin{aligned}
\mathbf{u}_L &= \sum_{k=1}^{n_L} (\mathbf{A}_L \cdot \mathbf{u}_L, \mathbf{y}_k) \mathbf{y}_k \\
\mathbf{M}_1^{\mathrm{T}} \mathbf{A}_L \mathbf{M}_1 \cdot \mathbf{u}_L &= \sum_{k=1}^{n_L} (\mathbf{A}_L \cdot \mathbf{u}_L, \mathbf{y}_k) \mathbf{M}_1^{\mathrm{T}} \mathbf{A}_L \mathbf{M}_1 \cdot \mathbf{y}_k = \sum_{k=1}^{n_L} (\mathbf{A}_L \cdot \mathbf{u}_L, \mathbf{y}_k) \lambda_k \mathbf{A}_L \cdot \mathbf{y}_k \\
\mathbf{u}_L^{\mathrm{T}} \cdot \mathbf{M}_1^{\mathrm{T}} \mathbf{A}_L \mathbf{M}_1 \cdot \mathbf{u}_L &= \sum_{k=1}^{n_L} \lambda_k (\mathbf{A}_L \cdot \mathbf{u}_L, \mathbf{y}_k)^2 \leq (\max_{1 \leq i \leq n_L} \lambda_i) \sum_{k=1}^{n_L} (\mathbf{A}_L \cdot \mathbf{u}_L, \mathbf{y}_k)^2 = (\max_{1 \leq i \leq n_L} \lambda_i) \mathbf{u}_L^{\mathrm{T}} \cdot \mathbf{A}_L \cdot \mathbf{u}_L.
\end{aligned} \tag{4.21}$$

Hence $\quad (\mathbf{M}^{-1}\mathbf{A} \cdot \mathbf{u})^{\mathrm{T}} \cdot \underline{\mathbf{A}}_L \cdot (\mathbf{M}^{-1}\mathbf{A} \cdot \mathbf{u}) \leq (\max_{1 \leq i \leq n_L} \lambda_i) \mathbf{u}^{\mathrm{T}} \cdot \underline{\mathbf{A}}_L \cdot \mathbf{u}.$ □

Theorem 4.1. *Let* $\mathbf{A} = \begin{pmatrix} \mathbf{A}_L & \mathbf{A}_{LR} \\ \mathbf{A}_{RL} & \mathbf{A}_R \end{pmatrix}$ *be an SPD matrix, and* $\underline{\mathbf{A}}_L$ *be defined as in* (4.16)*. For any invertible matrix* $\mathbf{M}^{-1}$ *we have*

$$\exists \tau > 0, \quad \forall \mathbf{u} \in \mathbb{R}^n : \quad ((\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}) \cdot \mathbf{u})^{\mathrm{T}} \cdot \underline{\mathbf{A}}_L \cdot ((\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}) \cdot \mathbf{u}) \leq \tau \mathbf{u}^{\mathrm{T}} \cdot \underline{\mathbf{A}}_L \cdot \mathbf{u}$$

if and only if

$$\exists \mathbf{W}_1 \in \mathbb{R}^{n_L \times n_L}, \mathbf{W}_3 \in \mathbb{R}^{n_R \times n_L}, \mathbf{W}_4 \in \mathbb{R}^{n_R \times n_R} : \mathbf{M}^{-1} = \begin{pmatrix} \mathbf{W}_1 & -\mathbf{W}_1 \mathbf{A}_{LR} \mathbf{A}_R^{-1} \\ \mathbf{W}_3 & \mathbf{W}_4 \end{pmatrix}. \tag{4.22}$$

Proof. According to Corollary 4.2, Lemmas 4.6 and 4.7, we have:

$$\left[\exists \tau > 0, \forall \mathbf{u} \in \mathbb{R}^n : ((\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}) \cdot \mathbf{u})^{\mathrm{T}} \cdot \underline{\mathbf{A}}_L \cdot ((\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}) \cdot \mathbf{u}) \leq \tau \mathbf{u}^{\mathrm{T}} \cdot \underline{\mathbf{A}}_L \cdot \mathbf{u}\right] \iff \left[\mathbf{R}_1 \mathbf{M}^{-1} \mathbf{A} \mathbf{R}_2^{\mathrm{T}} = \mathbf{0}\right];$$

where $\mathbf{R}_1 \in \mathbb{R}^{n_L \times n}, \mathbf{R}_2 \in \mathbb{R}^{n_R \times n}$ are restriction matrices defined such that $\mathbf{R}_1 : \begin{pmatrix} \mathbf{x}_L \\ \mathbf{x}_R \end{pmatrix} \in \mathbb{R}^n \mapsto \mathbf{x}_L \in \mathbb{R}^{n_L}$, and $\mathbf{R}_2 : \begin{pmatrix} \mathbf{x}_L \\ \mathbf{x}_R \end{pmatrix} \in \mathbb{R}^n \mapsto \mathbf{x}_R \in \mathbb{R}^{n_R}$. Let $\mathbf{W}_1 \in \mathbb{R}^{n_L \times n_L}, \mathbf{W}_2 \in \mathbb{R}^{n_L \times n_R}, \mathbf{W}_3 \in \mathbb{R}^{n_R \times n_L}, \mathbf{W}_4 \in \mathbb{R}^{n_R \times n_R}$ such that $\mathbf{M}^{-1} = \begin{pmatrix} \mathbf{W}_1 & \mathbf{W}_2 \\ \mathbf{W}_3 & \mathbf{W}_4 \end{pmatrix}$.

Then $\mathbf{R}_1 \mathbf{M}^{-1} = \begin{pmatrix} \mathbf{W}_1 & \mathbf{W}_2 \end{pmatrix}$ and $\mathbf{A}\mathbf{R}_2^{\mathrm{T}} = \begin{pmatrix} \mathbf{A}_{LR} \\ \mathbf{A}_R \end{pmatrix}$. Hence:

$$\mathbf{R}_1 \mathbf{M}^{-1} \mathbf{A} \mathbf{R}_2^{\mathrm{T}} = \mathbf{0} \iff \mathbf{W}_1 \mathbf{A}_{LR} + \mathbf{W}_2 \mathbf{A}_R = \mathbf{0} \iff \mathbf{W}_2 = -\mathbf{W}_1 \mathbf{A}_{LR} \mathbf{A}_R^{-1}.$$

□

Remark 4.1. Note that in the proof of Lemma 4.7, Inequality (4.21) is sharp, therefore the minimum coefficient τ that satisfies the property in Theorem 4.1 is the maximum eigenvalue λ_k of (4.20).

Remark 4.2. On another hand, it is worth observing that Theorem 4.1 does not impose any special requirements on matrices $\mathbf{W}_1$, $\mathbf{W}_3$ and $\mathbf{W}_4$. As a consequence, it is up to the user to set those blocks in a convenient way. For instance, the choice of a matrix $\mathbf{W}_3 := -\mathbf{A}_R^{-1}\mathbf{A}_{RL}\mathbf{W}_1^{\mathrm{T}}$ and symmetric diagonal blocks $\mathbf{W}_1$ and $\mathbf{W}_4$ may be considered for the purposes of symmetry. In this case, it is possible to use $\mathbf{M}^{-1}$ as a preconditioner outside the framework of fixed-point iteration schemes, for a PCG solver for example.

As far as the block $\mathbf{W}_4$ is concerned, we can choose it to be equal to $\mathbf{A}_R^{-1}$ as this inverse is already needed for an off-diagonal block.

Furthermore, if we denote $\underline{\mathbf{S}} := \mathbf{A}_L - \mathbf{A}_{LR}\mathbf{A}_R^{-1}\mathbf{A}_{RL}$ we can deduce the expression of the upper left block $\mathbf{M}_1$ of the matrix $\mathbf{M}^{-1}\mathbf{A}$ from (4.22) in Theorem 4.1 and the generalized eigenvalue problem (4.20) becomes:

$$\text{find} \quad (\lambda_k, \mathbf{y}_k) \in \mathbb{R} \times \mathbb{R}^{n_L} \quad \text{such that} \quad \underline{\mathbf{S}}\mathbf{W}_1^{\mathrm{T}}\mathbf{A}_L\mathbf{W}_1\underline{\mathbf{S}} \cdot \mathbf{y}_k = \lambda_k \mathbf{A}_L \cdot \mathbf{y}_k; \tag{4.23}$$

We are interested in the maximum λ_k as it gives the value of τ, i.e. the minimum upperbound of the error's growth rate. It is straightforward that the eigenvalues λ_k form the spectrum of the matrix $\underline{\mathbf{S}}\mathbf{W}_1^{\mathrm{T}}\mathbf{A}_L\mathbf{W}_1\underline{\mathbf{S}}\mathbf{A}_L^{-1}$. To render this spectrum bounded from above, three options can be considered:

❖ $\mathbf{W}_1 = \underline{\mathbf{S}}^{-1}$: This choice reduces the spectrum of $\underline{\mathbf{S}}\mathbf{W}_1^{\mathrm{T}}\mathbf{A}_L\mathbf{W}_1\underline{\mathbf{S}}\mathbf{A}_L^{-1}$ to 1 since

$$\underline{\mathbf{S}}\mathbf{W}_1^{\mathrm{T}}\mathbf{A}_L\mathbf{W}_1\underline{\mathbf{S}} = \mathbf{A}_L.$$

❖ $\mathbf{W}_1 = \mathbf{A}_L^{-1}$: This choice makes the spectrum of $\underline{\mathbf{S}}\mathbf{W}_1^{\mathrm{T}}\mathbf{A}_L\mathbf{W}_1\underline{\mathbf{S}}\mathbf{A}_L^{-1}$ bounded from above by 1. In fact,

$$\underline{\mathbf{S}}\mathbf{W}_1^{\mathrm{T}}\mathbf{A}_L\mathbf{W}_1\underline{\mathbf{S}}\mathbf{A}_L^{-1} = (\underline{\mathbf{S}}\mathbf{A}_L^{-1})^2;$$

And $\mathrm{Sp}(\underline{\mathbf{S}}\mathbf{A}_L^{-1}) \subset]0, 1]$ (see the proof of Lemma 3.1).

❖ Taking $\mathbf{W}_1$ as a SPD preconditioner such that the eigenvalues of $\mathbf{W}_1\mathbf{A}_L$ are below a fixed scalar $\nu > 0$, implies that the maximum λ_k (and hence the minimal τ) is less than ν^2 (see Lemma 4.8 below).

It should be outlined that the first two choices are more theoretical than practical as it is often costly to compute the exact inverse of a large block $\mathbf{A}_L$ or a dense Schur complement matrix $\underline{\mathbf{S}}$. The third choice is more affordable in practice, as there are many preconditioning strategies to bound the maximum eigenvalue of the preconditioned operator, e.g. domain decomposition-based preconditioners (like one-level Additive Schwarz, BDD or two-level Schwarz) [47] or LORASC [67].

Lemma 4.8. *Let $\mathbf{W}_1$ be a SPD preconditioner of $\mathbf{A}_L$ such that for a fixed scalar $\nu > 0$, we have*

$$\lambda_{\max}(\mathbf{W}_1\mathbf{A}_L) \leq \nu; \tag{4.24}$$

then it holds that:

$$\lambda_{\max}(\underline{\mathbf{S}}\mathbf{W}_1\mathbf{A}_L\mathbf{W}_1\underline{\mathbf{S}}\mathbf{A}_L^{-1}) \leq \nu^2. \tag{4.25}$$

the eigenvalues of $\mathbf{W}_1\mathbf{A}_L$ *are below a fixed scalar* $\nu > 0$*.*

Proof. We know that

$$\underline{\mathbf{S}}\mathbf{W}_1 = \mathbf{A}_R\mathbf{W}_1 - \mathbf{A}_{LR}\mathbf{A}_R^{-1}\mathbf{A}_{RL}\mathbf{W}_1$$

Therefore since $\mathbf{W}_1$ and $\mathbf{A}_{LR}\mathbf{A}_R^{-1}\mathbf{A}_{RL}$ are SPD matrices, the eigenvalues of $\mathbf{A}_{LR}\mathbf{A}_R^{-1}\mathbf{A}_{RL}\mathbf{W}_1$ are nonnegative therefore

$$\lambda_{\max}(\underline{\mathbf{S}}\mathbf{W}_1) \leq \lambda_{\max}(\mathbf{A}_L\mathbf{W}_1).$$

And also from the proof of Lemma 3.1:

$$\lambda_{\max}(\underline{\mathbf{S}}\mathbf{A}_L^{-1}) \leq 1.$$

Thus

$$\begin{aligned}\lambda_{\max}(\underline{\mathbf{S}}\mathbf{W}_1\mathbf{A}_L\mathbf{W}_1\underline{\mathbf{S}}\mathbf{A}_L^{-1}) &\leq \lambda_{\max}(\underline{\mathbf{S}}\mathbf{W}_1)\lambda_{\max}(\mathbf{A}_L\mathbf{W}_1)\lambda_{\max}(\underline{\mathbf{S}}\mathbf{A}_L^{-1}) \\ &\leq \lambda_{\max}^2(\mathbf{A}_L\mathbf{W}_1) \\ &\leq \nu^2.\end{aligned}$$

□

4.6 Link with the adaptive preconditioner for PCG based on local error indicators

In this section, we consider the preconditioner introduced in Section 2.5 of Chapter 2, and make the link with the preconditioner suggested in Section 4.5. Indeed, we exploit the lemmas proven in this latter section to derive the properties satisfied by the adaptive preconditioner introduced in Chapter 2.

First, we start by the preconditioner $\mathbf{M} = \begin{pmatrix} \mathbf{A}_L & \mathbf{A}_{LR} \\ \mathbf{A}_{RL} & \mathbf{M}_S + \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR} \end{pmatrix}$ suggested in Theorem 2.1, where $\mathbf{M}_S$ is a preconditioner for $\mathbf{S} := \mathbf{A}_R - \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR}$. That theorem states that when preconditioned by this preconditioner, the PCG solver on the whole system becomes equivalent to a PCG solver on a reduced Schur complement system resulting in a nil residual on the nodes of Ω_1 (i.e. the L-part) at each iteration:

$$\forall i > 0, \quad \mathbf{R}_1 \cdot \mathbf{r}^{(i)} = \mathbf{R}_1\mathbf{A} \cdot (\mathbf{x} - \mathbf{x}^{(i)}) = \mathbf{0}, \tag{4.26}$$

where $\mathbf{R}_1$ is the restriction matrix of (4.5). In the following lemma, we prove that this property is still satisfied with a fixed-point iteration scheme (4.12).

Lemma 4.9. *Let* $\mathbf{M} = \begin{pmatrix} \mathbf{A}_L & \mathbf{A}_{LR} \\ \mathbf{A}_{RL} & \mathbf{M}_S + \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR} \end{pmatrix}$ *where* $\mathbf{M}_S$ *is invertible. When* $\mathbf{M}^{-1}$ *is used as the preconditioner of the fixed-point iteration scheme defined in* (4.12)*, then property* (4.26) *holds regardless of the choice of the initial guess* $\mathbf{x}^{(0)}$*.*

Proof. The inverse of $\mathbf{M}$ is expressed as

$$\mathbf{M}^{-1} = \begin{pmatrix} \mathbf{A}_L^{-1} + \mathbf{A}_L^{-1}\mathbf{A}_{LR}\mathbf{M}_S^{-1}\mathbf{A}_{RL}\mathbf{A}_L^{-1} & -\mathbf{A}_L^{-1}\mathbf{A}_{LR}\mathbf{M}_S^{-1} \\ -\mathbf{M}_S^{-1}\mathbf{A}_{RL}\mathbf{A}_L^{-1} & \mathbf{M}_S^{-1} \end{pmatrix}$$

Therefore,

$$\mathbf{M}^{-1}\mathbf{A} = \begin{pmatrix} \mathbf{I} & \mathbf{A}_L^{-1}\mathbf{A}_{LR}(\mathbf{I} - \mathbf{M}_S^{-1}\mathbf{S}) \\ \mathbf{0} & \mathbf{M}_S^{-1}\mathbf{S} \end{pmatrix}$$
$$\mathbf{A}(\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}) = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} - \mathbf{S}\mathbf{M}_S^{-1}\mathbf{S} \end{pmatrix}.$$

Thus, $\mathbf{R}_1\mathbf{A}(\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}) = \mathbf{0}$
Consequently, for any iteration $i > 0$ we have

$$\mathbf{R}_1\mathbf{r}^{(i)} = \mathbf{R}_1\mathbf{A} \cdot (\mathbf{x} - \mathbf{x}^{(i)}) = \mathbf{R}_1\mathbf{A}(\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}) \cdot (\mathbf{x} - \mathbf{x}^{(i-1)}) = \mathbf{0}$$

□

For the more general shape of the adaptive preconditioner proposed in Theorem 2.2, we prove in what follows that Property (4.26) still holds but this time for a specific initial guess $\mathbf{x}^{(0)}$.

Lemma 4.10. *Let $\mathbf{x}_R^{(0)}$ be an arbitrary vector of $\mathbb{R}^{n_R}$ and $\mathbf{W}_1 \in \mathbb{R}^{n_L \times n_L}$, $\mathbf{W}_2 \in \mathbb{R}^{n_R \times n_R}$ two invertible matrices. Let the linear system $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ be solved by a fixed-point iteration scheme (4.12) with a preconditioner $\mathbf{M} = \mathbf{W}\mathbf{W}^{\mathrm{T}}$ and an initial guess $\mathbf{x}^{(0)}$ such that*

$$\mathbf{x}^{(0)} = \begin{bmatrix} \mathbf{A}_L^{-1} \cdot (\mathbf{b}_L - \mathbf{A}_{LR} \cdot \mathbf{x}_R^{(0)}) \\ \mathbf{x}_R^{(0)} \end{bmatrix}, \quad \mathbf{W} = \begin{pmatrix} \mathbf{W}_1 & \mathbf{0} \\ \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{W}_1 & \mathbf{W}_2 \end{pmatrix}; \tag{4.27}$$

then $\mathbf{R}_1 \cdot (\mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(i)}) = \mathbf{0}$ at each iteration $i > 0$.

Proof. The definition of $\mathbf{x}^{(0)}$ yields

$$\mathbf{x} - \mathbf{x}^{(0)} = \begin{bmatrix} -\mathbf{A}_L^{-1}\mathbf{A}_{LR} \cdot (\mathbf{x}_R - \mathbf{x}_R^{(0)}) \\ \mathbf{x}_R - \mathbf{x}_R^{(0)} \end{bmatrix} = \begin{pmatrix} -\mathbf{A}_L^{-1}\mathbf{A}_{LR} \\ \mathbf{I} \end{pmatrix} \cdot (\mathbf{x}_R - \mathbf{x}_R^{(0)}).$$

We know that due to (4.12), we have for any i,

$$\mathbf{R}_1 \cdot (\mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(i)}) = \mathbf{R}_1\mathbf{A} \cdot (\mathbf{x} - \mathbf{x}^{(i)}) = \mathbf{R}_1\mathbf{A}(\mathbf{I} - \mathbf{M}^{-1}\mathbf{A})^i \cdot (\mathbf{x} - \mathbf{x}^{(0)}).$$

Therefore, proving the lemma amounts to prove that

$$\forall i \in \mathbb{N}, \quad \mathbf{R}_1\mathbf{A}(\mathbf{I} - \mathbf{M}^{-1}\mathbf{A})^i \begin{pmatrix} -\mathbf{A}_L^{-1}\mathbf{A}_{LR} \\ \mathbf{I} \end{pmatrix} = \mathbf{0}. \tag{4.28}$$

Let us demonstrate that by induction.

For $i = 0$,

$$\mathbf{R}_1\mathbf{A}\begin{pmatrix}-\mathbf{A}_L^{-1}\mathbf{A}_{RL}\\ \mathbf{I}\end{pmatrix} = \begin{pmatrix}\mathbf{A}_L & \mathbf{A}_{LR}\end{pmatrix}\begin{pmatrix}-\mathbf{A}_L^{-1}\mathbf{A}_{RL}\\ \mathbf{I}\end{pmatrix} = \mathbf{0}.$$

We denote $\mathbf{M}_1 := \mathbf{W}_1\mathbf{W}_1^{\mathrm{T}}$, and $\mathbf{M}_S := \mathbf{W}_2\mathbf{W}_2^{\mathrm{T}}$. A quick computation of the inverse of $\mathbf{W}$ gives

$$\begin{aligned}\mathbf{M}^{-1} = \mathbf{W}^{-\mathrm{T}}\mathbf{W}^{-1} &= \begin{pmatrix}\mathbf{M}_1^{-1} + \mathbf{A}_L^{-1}\mathbf{A}_{LR}\mathbf{M}_S^{-1}\mathbf{A}_{RL}\mathbf{A}_L^{-1} & -\mathbf{A}_L^{-1}\mathbf{A}_{LR}\mathbf{M}_S^{-1}\\ -\mathbf{M}_S^{-1}\mathbf{A}_{RL}\mathbf{A}_L^{-1} & \mathbf{M}_S^{-1}\end{pmatrix}\\ \mathbf{I} - \mathbf{M}^{-1}\mathbf{A} &= \begin{pmatrix}\mathbf{I} - \mathbf{M}_1^{-1}\mathbf{A}_L & -\mathbf{M}_1^{-1}\mathbf{A}_{LR} + \mathbf{A}_L^{-1}\mathbf{A}_{LR}\mathbf{M}_S^{-1}\mathbf{S}\\ \mathbf{0} & \mathbf{I} - \mathbf{M}_S^{-1}\mathbf{S}\end{pmatrix}\end{aligned} \tag{4.29}$$

Let $i \in \mathbb{N}$, we assume the induction hypothesis (4.28) for i, then due to (4.29) we have

$$\begin{aligned}\mathbf{R}_1\mathbf{A}(\mathbf{I} - \mathbf{M}^{-1}\mathbf{A})^{i+1}\begin{pmatrix}-\mathbf{A}_L^{-1}\mathbf{A}_{LR}\\ \mathbf{I}\end{pmatrix} &= \begin{pmatrix}\mathbf{A}_L & \mathbf{A}_{LR}\end{pmatrix}(\mathbf{I} - \mathbf{M}^{-1}\mathbf{A})^i(\mathbf{I} - \mathbf{M}^{-1}\mathbf{A})\begin{pmatrix}-\mathbf{A}_L^{-1}\mathbf{A}_{LR}\\ \mathbf{I}\end{pmatrix}\\ &= \begin{pmatrix}\mathbf{A}_L & \mathbf{A}_{LR}\end{pmatrix}(\mathbf{I} - \mathbf{M}^{-1}\mathbf{A})^i\begin{pmatrix}-\mathbf{A}_L^{-1}\mathbf{A}_{LR}(\mathbf{I} - \mathbf{M}_S^{-1}\mathbf{S})\\ (\mathbf{I} - \mathbf{M}_S^{-1}\mathbf{S})\end{pmatrix}\\ &= \begin{pmatrix}\mathbf{A}_L & \mathbf{A}_{LR}\end{pmatrix}(\mathbf{I} - \mathbf{M}^{-1}\mathbf{A})^i\begin{pmatrix}-\mathbf{A}_L^{-1}\mathbf{A}_{LR}\\ \mathbf{I}\end{pmatrix}(\mathbf{I} - \mathbf{M}_S^{-1}\mathbf{S})\\ &= 0.\end{aligned}$$

Thus, we can apply the induction hypothesis for i to show that (4.28) is true for $i+1$. □

Moreover, as far as the algebraic error norm is concerned, if we denote by

$$\underline{\mathbf{A}}_R := \begin{pmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{A}_R\end{pmatrix} \in \mathcal{M}_{n,n}(\mathbb{R}), \tag{4.30}$$

we can derive the subsequent corollary for these adaptive preconditioners:

Corollary 4.3. *Let $\mathbf{M}^{-1}$ be an adaptive preconditioner of the matrix $\mathbf{A}$ as defined in Lemma 4.10. When $\mathbf{M}^{-1}$ is used as the preconditioner of the fixed-point iteration scheme defined in (4.12), then it holds that:*

$$\exists\tau > 0, \quad \forall\mathbf{u} \in \mathbb{R}^n : \quad (\mathbf{M}^{-1}\mathbf{A}\cdot\mathbf{u})^{\mathrm{T}}\cdot\underline{\mathbf{A}}_R\cdot(\mathbf{M}^{-1}\mathbf{A}\cdot\mathbf{u}) \leq \tau\mathbf{u}^{\mathrm{T}}\cdot\underline{\mathbf{A}}_R\cdot\mathbf{u}.$$

The minimum value of τ is the maximum eigenvalue of the generalized eigenvalue problem:

$$\textit{find} \quad (\lambda_k, \mathbf{y}_k) \in \mathbb{R}\times\mathbb{R}^{n_R} \quad \textit{such that} \quad \mathbf{M}_4^{\mathrm{T}}\mathbf{A}_R\mathbf{M}_4\cdot\mathbf{y}_k = \lambda_k\mathbf{A}_R\cdot\mathbf{y}_k; \tag{4.31}$$

where $\mathbf{M}_4$ is the bottom right block of size $n_R \times n_R$ of the matrix $\mathbf{M}^{-1}\mathbf{A}$.

Proof. First, if we look at the shape of the adaptive preconditioners either in its general shape (as in Lemma 4.10) or in its particular shape (as in Lemma 4.9) we notice that the preconditioned

operator takes the following block shape:

$$\exists(\underline{\mathbf{M}}_1, \mathbf{M}_3, \mathbf{M}_4) \in \mathcal{M}_{n_L,n_L}(\mathbb{R}) \times \mathcal{M}_{n_L,n_R}(\mathbb{R}) \times \mathcal{M}_{n_R,n_R}(\mathbb{R}) : \mathbf{M}^{-1}\mathbf{A} = \begin{pmatrix} \underline{\mathbf{M}}_1 & \mathbf{M}_3 \\ \mathbf{0} & \mathbf{M}_4 \end{pmatrix}$$

Therefore, we can apply Lemma 4.7 by switching or commuting the roles of the subsets L and R. In this case, the result follows immediately. □

Now, let us summarize and comment the results of the corollaries of this section. We have proven that the main feature of adaptive preconditioner introduced for PCG solver in Chapter 2, that is the projection of the residual on the nodes of subdomain Ω_1 (L-part) is nil at each iteration, is still valid with a fixed-point iteration scheme. We also proved that the adaptive preconditioner satisfies with the same scheme a criterion that expresses that the $\underline{\mathbf{A}}_R$-seminorm of the error does not increase more than τ-times from iteration i to iteration $i+1$. Of course, the discussion over the choices of the diagonal blocks of the preconditioner in the end of Section 4.5 holds for this part as well. On another note, there is no question that this $\underline{\mathbf{A}}_R$-seminorm of the error represents a share of the global energy norm of the error. The evolution of this share from an iteration to another is controlled by the adaptive preconditioner and the growth rate is kept under a fixed threshold. That being said, the starting hypothesis (4.6) does not enable us to determine whether this share is dominant. Indeed, (4.6) expresses that the local algebraic error on subdomain Ω_1, which is the $\mathbf{A}_p^{(1)}$-seminorm of the error, is dominant over the local algebraic error on subdomain Ω_2, which is the $\mathbf{A}_p^{(2)}$-seminorm of the error. Yet, in general we cannot prove any partial order or superiority relationship, in the sense of matrix positiveness, between the matrices $\underline{\mathbf{A}}_R$ and $\mathbf{A}_p^{(2)}$ as we did between the matrices $\underline{\mathbf{A}}_L$ and $\mathbf{A}_p^{(1)}$. Indeed, we demonstrate that with the following counterexample. We consider a Poisson's equation with Dirichlet boundary conditions on the square $[0,1] \times [0,1]$. We discretize this PDE by FEM on the uniform grid shown in Figure 4.1. Note that with FEM, the boundary conditions are taken into account in the evaluation of the right

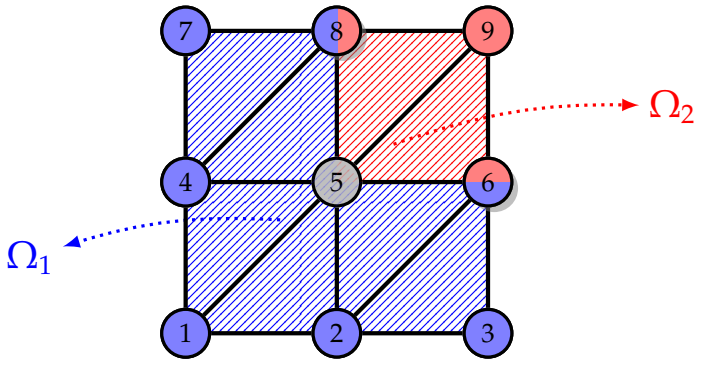


Figure 4.1 – *Simple example of the decomposition with a* 2×2 *mesh grid.*

hand side vector of the linear system. The global matrix obtained for the natural ordering of the

degrees of freedom (from 1 to 9) is:

$$\mathbf{A} = \begin{pmatrix} 1 & -1/2 & 0 & -1/2 & 0 & 0 & 0 & 0 & 0 \\ -1/2 & 2 & -1/2 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1/2 & 1 & 0 & 0 & -1/2 & 0 & 0 & 0 \\ -1/2 & 0 & 0 & 2 & -1 & 0 & -1/2 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1/2 & 0 & -1 & 2 & 0 & 0 & -1/2 \\ 0 & 0 & 0 & -1/2 & 0 & 0 & 1 & -1/2 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1/2 & 2 & -1/2 \\ 0 & 0 & 0 & 0 & 0 & -1/2 & 0 & -1/2 & 1 \end{pmatrix}$$

If we consider the domain decomposition illustrated in Figure 4.1, then the set of interior nodes of subdomain Ω_2 comprises only the node 9 whereas the nodes 5, 6 and 8 are in the interface. Therefore, if we keep the natural ordering then the matrices $\underline{\mathbf{A}}_R$ and $\mathbf{A}_p^{(2)}$ are equal to:

$$\underline{\mathbf{A}}_R = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{A}_p^{(2)} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1/2 & 0 & -1/2 & 0 \\ 0 & 0 & 0 & 0 & -1/2 & 1 & 0 & 0 & -1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1/2 & 0 & 0 & 1 & -1/2 \\ 0 & 0 & 0 & 0 & 0 & -1/2 & 0 & -1/2 & 1 \end{pmatrix}$$

Therefore, the difference is:

$$\mathbf{A}_p^{(2)} - \underline{\mathbf{A}}_R = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1/2 & 0 & -1/2 & 0 \\ 0 & 0 & 0 & 0 & -1/2 & 1 & 0 & 0 & -1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1/2 & 0 & 0 & 1 & -1/2 \\ 0 & 0 & 0 & 0 & 0 & -1/2 & 0 & -1/2 & 0 \end{pmatrix}$$

And its eigenvalues can be computed: $\lambda_1 \approx -0.45161$; $\lambda_2 = 0$; $\lambda_3 \approx 0.59697$; $\lambda_4 = 1$; $\lambda_5 \approx 1.8546$. We can notice that they are not all nonnegative.

Furthermore, it should be stressed that the properties of the adaptive preconditioner introduced in Theorem 2.1 of Chapter 2, and the preconditioner introduced in Lemma 4.1 of this chapter are complementary, in the sense that each allows to control a share of the global energy norm of the error. The first one requires inverting the L-block corresponding to a subdomain with a high algebraic error, reduces the $\underline{\mathbf{A}}_R$-seminorm of the error and cancels the residual on the nodes of L, whereas the second one requires inverting the R-block corresponding to a subdomain with low algebraic error and ensures the growth rate of the $\underline{\mathbf{A}}_L$-seminorm of the error stays below a given threshold. Therefore, the choice between those two depends on the size of submatrix $\mathbf{A}_R$ with respect to $\mathbf{A}_L$. Note also that the exact inverse of the matrix $\mathbf{A}$ satisfies the shapes of both preconditioners.

4.7 Numerical results

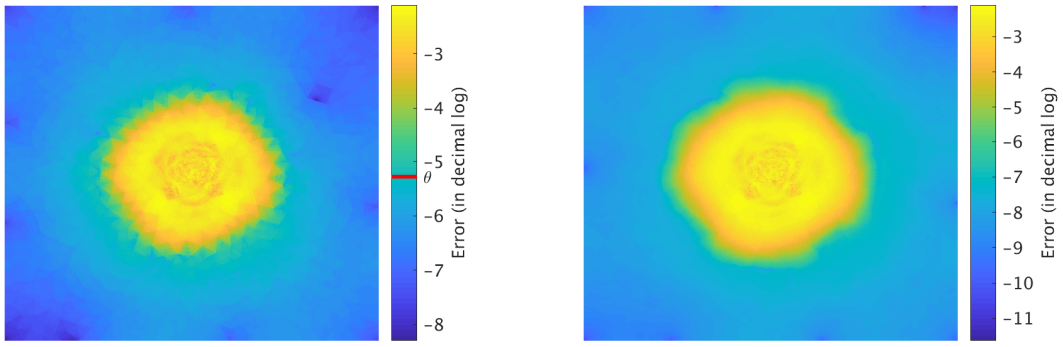
In this section, we consider the experimental framework of Section 2.6 of Chapter 2, and reuse the test cases and configuration described in Section 2.6.2. The tests are carried in Matlab. We generate a uniform mesh and use PDE toolbox to solve the considered PDE on the domain $\Omega = [-1, 1]$. Once the linear system is built, we run a few iterations of the linear solver (20 iterations of PCG preconditioned by a Block-Jacobi preconditioner) to get an initial estimation of the algebraic error on all the elements of the mesh. From the values of these a posteriori error estimates, we determine the subsets of indices L and R.

We check first with a fixed-point iteration scheme that the result of Theorem 4.1 is applicable in practice and that the growth rate of the $\underline{\mathbf{A}}_L$-seminorm of the error is controlled by the choice of the block $\mathbf{W}_1$ in the preconditioner defined in (4.22). We denote this preconditioner as the L-adaptive preconditioner. We would like to mention that the L-adaptive preconditioner implemented in our Matlab prototype is nested (see [100]) in the sense that it does not build the inverse of the block $\mathbf{A}_R$ but the application of the preconditioner is carried out by a call to an inner solver (PCG in our configuration) with a reduced relative tolerance of 10^{-2} and a reduced maximum number of iterations (set to 100). As a preconditioner for the inner solve, we reuse the initial Block-Jacobi preconditioner restricted to the R-indices. For checking purposes, we consider two different preconditioners for the block $\mathbf{W}_1$ for which an upper bound for the eigenvalues of the preconditioned operator is known: a LORASC preconditioner and a Block-Jacobi preconditioner for $\mathbf{A}_L$. For the latter one, we vary the number of diagonal blocks (2, 4, or 8).

Second, we test the L-adaptive preconditioner with PCG solver and compare its results to the preconditioner introduced in Chapter 2, that we denote as the R-adaptive preconditioner. For the numerical experiments with PCG, we consider a fixed block-size (≈ 5000) for the diagonal blocks used to build Block-Jacobi preconditioners, and a stopping threshold value of 10^{-6} for the euclidean norm of the residual.

The first three test cases of Section 2.6 in Chapter 2 are reused here. As far as the mesh configuration is concerned, we take an uniform mesh with maximum edge size $H_{max} = 0.1$ (for the first two test cases) and $H_{max} = 0.01$ (for the third test case), and the total number of elements is equal to 87 552 (for the first two test cases) and 32 544 (for the third test case). After discretization, the size of the

matrix $\mathbf{A}$ is $43\,457 \times 43\,457$ (for the first two test cases) and $16\,057 \times 16\,057$ (for the third test case).



(a) Algebraic a posteriori error estimates after 20 iterations

(b) Algebraic errors after 20 iterations

Figure 4.2 – *Initial distribution and a posteriori estimation of algebraic error for test case n°1*

For the first test case, the initial distribution and a posteriori estimation of algebraic error (after $j_0 = 20$ iterations) over the domain Ω are shown in Figure 4.2 where the color shades selected in the subdomain Ω_1 are indicated by the red marker on the left subfigure.

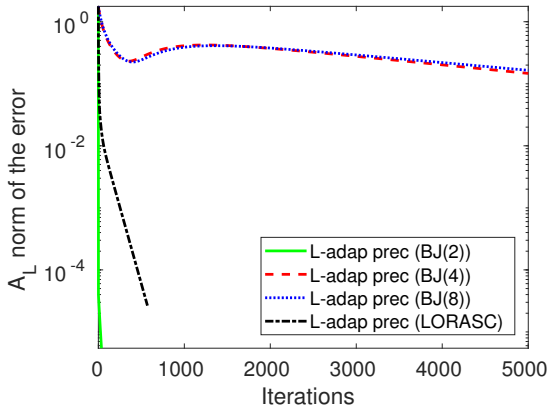
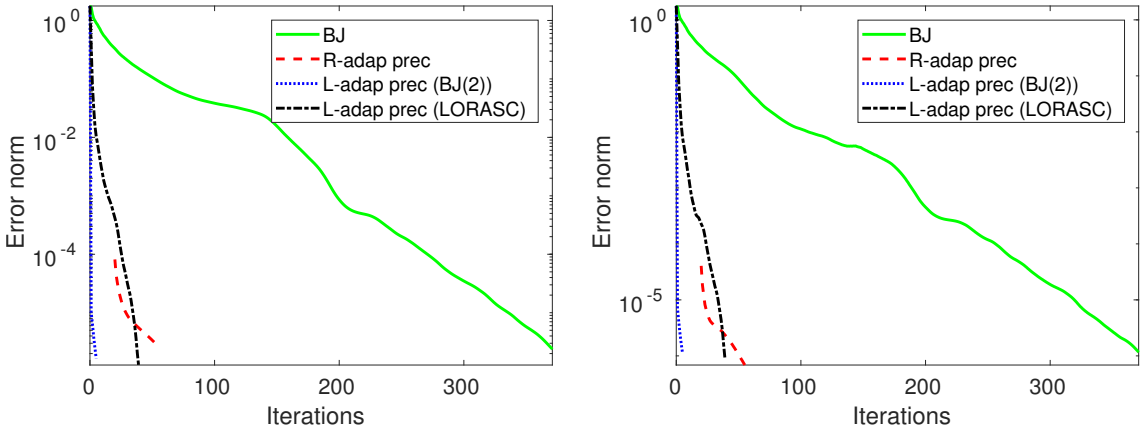


Figure 4.3 – *Evolution of the $\underline{\mathbf{A}}_L$-seminorm of the error with a fixed-point iteration scheme for test case n°1*

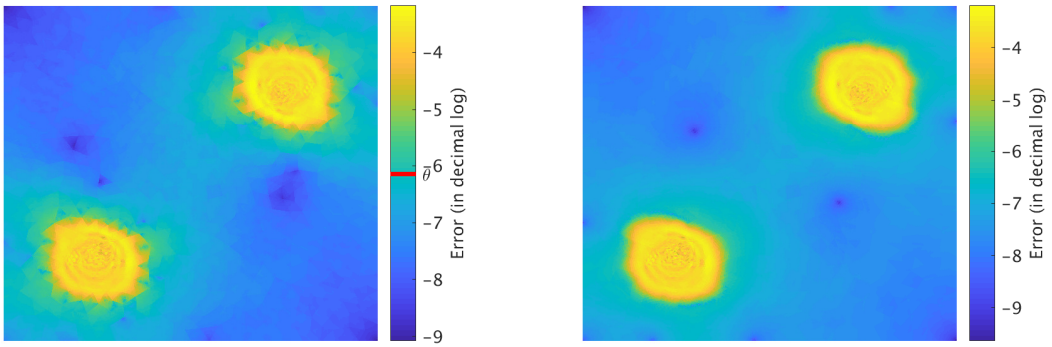
Figure 4.3 displays the $\underline{\mathbf{A}}_L$-seminorm of the error when using a fixed-point iteration scheme preconditioned by four configurations of the L-adaptive preconditioner, whereas the evolution of the global energy norm and the L-norm of the error when using a PCG solver preconditioned by a Block-Jacobi preconditioner, the R-adaptive preconditioner and the L-adaptive preconditioner is represented in Figure 4.4.

Likewise, we present the results obtained for the second test case. Figure 4.5 displays the initial error distribution over the mesh.



(a) Energy norm of the algebraic error (b) L-norm of the algebraic error

Figure 4.4 – *Error evolution with PCG for test case n°1*



(a) Algebraic a posteriori error estimates after 20 iterations (b) Algebraic errors after 20 iterations

Figure 4.5 – *Initial distribution and a posteriori estimation of algebraic error for test case n°2*

Figure 4.6 shows the $\underline{\mathbf{A}}_L$-seminorm of the error when using a fixed-point iteration scheme with the L-adaptive preconditioner. The global energy norm and the L-norm of the error are plotted in Figure 4.7 for the PCG solves with the L-adaptive and the R-adaptive preconditioners.

The results of the third test case are as following. Figure 4.8 displays the initial algebraic error distribution over the mesh, and the evolution of the $\underline{\mathbf{A}}_L$ norm of the error with a fixed-point iteration scheme preconditioned by L-adaptive preconditioners. Then the evolution of the global energy norm and the L-norm of the error when using a PCG solver preconditioned by a Block-Jacobi preconditioner, the R-adaptive preconditioner and the L-adaptive preconditioner is represented in Figure 4.9.

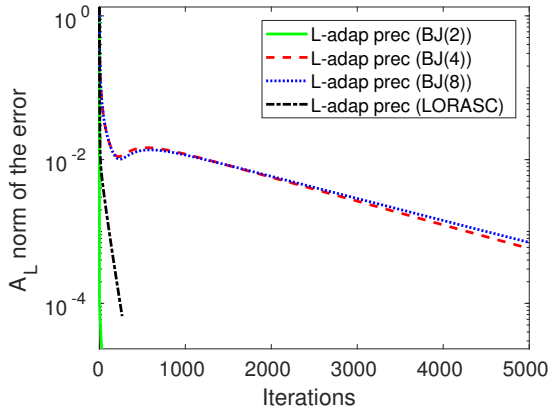
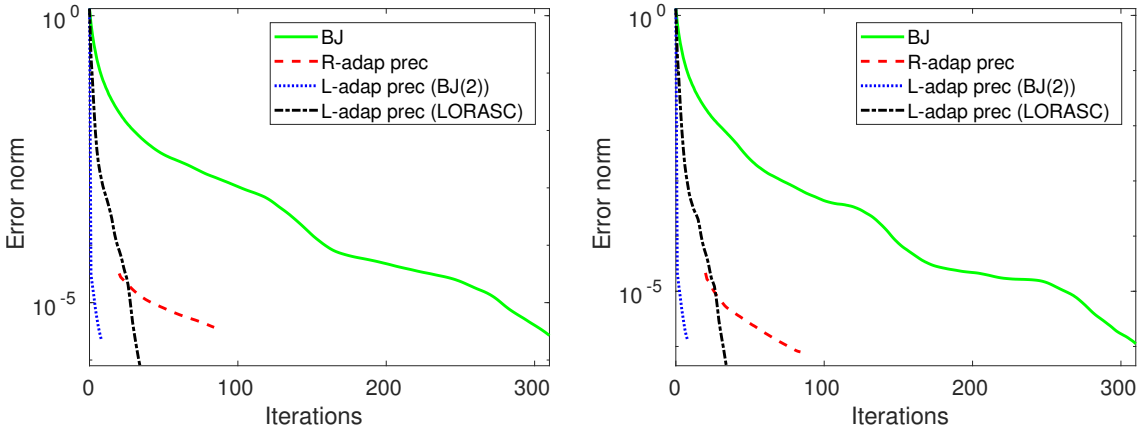


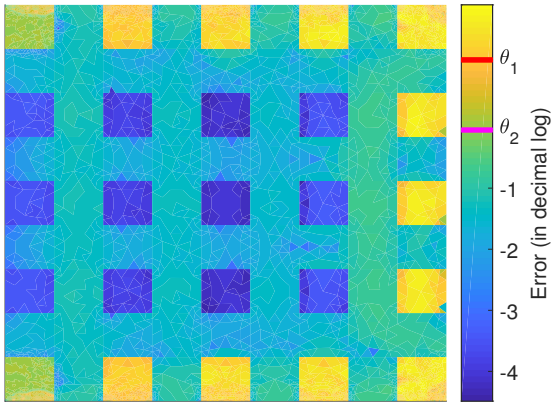
Figure 4.6 – *Evolution of the $\underline{\mathbf{A}}_L$-seminorm of the error with a fixed-point iteration scheme for test case n°2*



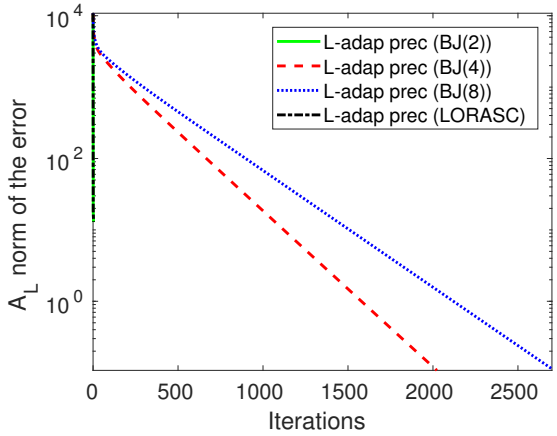
(a) Energy norm of the algebraic error

(b) L-norm of the algebraic error

Figure 4.7 – *Error evolution with PCG for test case n°2*

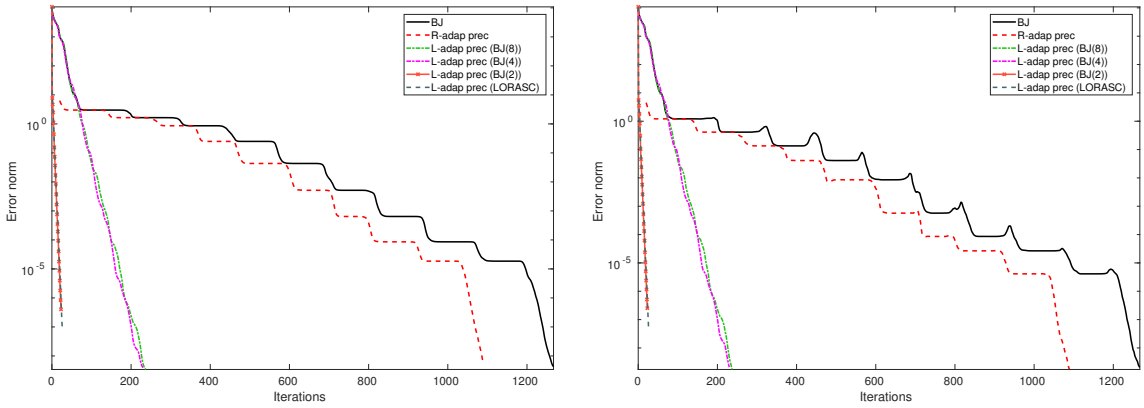


(a) Algebraic a posteriori error estimates after 20 iterations



(b) Evolution of the $\underline{\mathbf{A}}_L$-seminorm of the error

Figure 4.8 – *Initial distribution of algebraic error and evolution of the $\underline{\mathbf{A}}_L$-seminorm of the error with a fixed-point iteration scheme for test case n°3*



(a) Energy norm of the algebraic error (b) L-norm of the algebraic error

Figure 4.9 – *Error evolution with PCG for test case n°3*

Figures 4.3 and 4.6 show that the $\underline{\mathbf{A}}_L$-seminorm of the error is monotonically decreasing when a LORASC preconditioner or a two-block diagonal preconditioner is used to approximate $\mathbf{A}_L$. But it is not the case when we use a block diagonal preconditioner with more than two blocks.

This fact relates to ν, the upper bound of the largest eigenvalue of $\mathbf{W}_1\mathbf{A}_L$. Indeed, it follows from Lemma 4.8 that the growth rate of the $\underline{\mathbf{A}}_L$-seminorm of the error (i.e. the minimal τ in (4.18)) is less than or equal to ν^2. Lemma 0.1 and Lemma 3.2 yield the value of the bound ν obtained when $\mathbf{W}_1$ is taken as a m-block diagonal preconditioner and a LORASC preconditioner of $\mathbf{A}_L$ respectively. ν is equal to $1 + \max\limits_{1\leq i\leq m} \sum\limits_{\substack{j=1 \\ j\neq i}}^{m} \gamma_{ij}$ for the former and to 1 for the latter, where the CBS constants γ_{ij} are defined by (5) adapted to $\mathbf{A}_L$. Since $\mathbf{A}_L$ is positive definite, then the $\left(\gamma_{ij}\right)_{ij}$ are less than 1. In practice, and for the matrices tested in this section, a single CBS constant is very small, thus for a two-block diagonal preconditioner, the expression of ν contains only one CBS constant. As a consequence, $\nu^2 = (1+\gamma_{12})^2$ is, to a certain extent, close to 1, resulting in monotonic $\underline{\mathbf{A}}_L$-seminorm of the error. The same finding applies for the case when a LORASC preconditioner is selected for $\mathbf{W}_1$ as ν equals 1. However, for larger number of blocks (here for example $m \in \{4, 8\}$), ν comprises the sum of $m-1$ CBS constants, which is significant. Therefore, ν^2 greatly exceeds 1 and the $\underline{\mathbf{A}}_L$-seminorm of the error is not monotonic this time. In summary, the magnitude of ν^2 determines the rate of decline of the $\underline{\mathbf{A}}_L$-seminorm of the error. In other words, the growth rate τ is controlled by the quality of the preconditioner for the L-block. All this is reflected in Figures 4.3, 4.6 and 4.8.

Furthermore, one observes that those preconditioners, which we refer to as L-adap prec (BJ(2)) and L-adap prec (LORASC) in the graph legends above and for which the $\underline{\mathbf{A}}_L$-seminorm of the error is strictly decreasing in a fixed-point iteration scheme, perform well when a PCG solver is used as well (Tables 4.4, 4.7 and 4.9). Table 4.1 gives the number of iterations for the PCG solve with the L-adaptive preconditioners, with the R-adaptive preconditioner and with an initial Block-Jacobi preconditioner. We notice that even though the number of iterations of PCG is reduced when going from the initial BJ preconditioner to the R-adaptive preconditioner, we still

∗. The 20 iterations count here is due to the fact that the intermediate solution is used to compute the new initial guess for the R-adaptive preconditioner.

Table 4.1 – *The total number of iterations (IT) needed for the convergence of the preconditioned solve of the linear systems stemming from test cases* 1, 2 *and* 3*.*

	Test case n°1	**Test case n°2**	**Test case n°3**
initial BJ prec.	371	310	1 267
R-adap prec.*	36 (+20)	65 (+20)	1 069 (+20)
L-adap prec. (LORASC)	40	35	26
L-adap prec (BJ(2))	6	9	23

get an improvement by using the *L*-adaptive one, in both variants *L*-adap prec (LORASC) and *L*-adap prec (BJ(2)). In fact, the drop in the number of iterations is more important in the third test case where the *R*-adaptive preconditioner is not sufficient to significantly reduce the number of iterations, whereas the *L*-adaptive one manages to converge within around twenty iterations only. This can be due to the fact that the algebraic error is more scattered in this third test case, and that the size of $\mathbf{A}_L$ is almost twice that of $\mathbf{A}_R$ for this test case.

4.8 Conclusion

In this chapter, we have presented a second variant of an algebraic preconditioner that is designed to control the growth rate of the $\underline{\mathbf{A}}_L$-seminorm of the error when used within a fixed-point iteration scheme. Indeed, we have proven the relationship between that growth rate and the largest eigenvalue of the *L*-block of the preconditioned operator $\mathbf{M}^{-1}\mathbf{A}$. Furthermore, we also made the link with the first variant introduced in Chapter 2, and proved the properties satisfied for this shape of preconditioners in a fixed-point iteration scheme. Then, we tested the second variant in a PCG solver and compared it with the first variant. The results are encouraging with fewer iterations needed for convergence.

5

Application to some industrial simulations with finite volume discretization

Contents

Abstract

In this chapter, we apply the adaptive approach of Chapter 2 for iterative solution of linear systems arising from some industrial simulations. The numerical framework is different from that of Chapter 2, as we are now dealing with linear systems stemming from cell-centered finite volume discretization of single phase flow models. We adapt the starting assumption and consequently the solve procedure to the finite volume method. Numerical results of a reservoir simulation example for heterogeneous porous media in two dimensions are discussed. Using the adaptive solve procedure, we obtain a significant gain in terms of the number of time steps and iterations compared to a standard solve.
Keywords— single phase flow, finite volume method, Schur complement, steady/unsteady problems.

5.1 Introduction

In this chapter we propose an efficient adaptive solve procedure for solving linear systems stemming from finite-volume discretization of PDEs. For the ease of presentation, the following section starts by a steady model of single phase flow in porous media then an unsteady model of the same problem extends the scope of the experimentation (in Section 5.5.2.1).

This chapter is structured as follows. Section 5.2 introduces the steady problem of single phase flow in porous media with finite volume discretization, Section 5.3 formulates the starting hypothesis and presents some elements related to the field of a posteriori error analysis that justify the need for an adaptive solving procedure. Section 5.4 provides details of that error reducing approach. Finally, the numerical results of the procedure are shown in Section 5.5.

Some notations

Let $\Omega \subset \mathbb{R}^d$, $1 \leq d \leq 3$ be a polytopal domain (open, bounded and connected set). We denote by $\overline{\Omega}$, Ω^o, $\partial\Omega$ and $\mathcal{T}_h$ the closure, interior, boundary and a matching simplicial mesh of Ω, respectively. We consider that the domain Ω can be decomposed into n non-overlapping Lipschitz polyhedra K_i such that $K_i^o \cap K_j^o = \emptyset$ and $\bigcup_i \overline{K}_i = \overline{\Omega}$. For a polyhedron K_i, h_i designates its the diameter, $|K_i|$ its d-Lebesgue measure and $V(i)$ the set of indices of neighboring polyhedra. This tessellation of Ω is denoted $\mathcal{T}_h$ with $h = \max_i h_i$.

Let us denote by $\mathcal{F} = \mathcal{F}_{\text{int}} \cup \mathcal{F}_{\text{ext}}$ the set of mesh edges where $\mathcal{F}_{\text{int}}$ (resp. $\mathcal{F}_{\text{ext}}$) is the set of inner (resp. boundary) egdes. For a mesh element K_i, $\mathcal{F}_i$ is the set of all its edges, and for an edge $\sigma \in \mathcal{F}_i$, $|\sigma|$ denotes the $(d-1)$-Lebesgue measure of σ while $\boldsymbol{n}_{\sigma,i}$ and $d_{i,\sigma}$ respectively denote the unit normal vector to the edge σ and outward to K_i, and the distance between the cell center of K_i and the edge center of σ

We use the standard notation $L^2(\Omega)$, $H^1(\Omega)$ and $H_0^1(\Omega)$ for the spaces of integrable functions, resp. integrable functions admitting weak derivations, and trace vanishing on $\partial\Omega$. For a vector w of length $n \in \mathbb{N}$ and a subset $L \subset [\![1, n]\!]$, we denote by w_L the restriction of w to its components whose indexes belong to L.

5.2 Cell-centered finite volume discretization

Similar to Section 2.2 in Chapter 2 for finite element method, here we first introduce the steady single phase flow model problem in a simple cell-centered finite volume discretization, then define the resulting linear system of algebraic equations and present the key assumption that motivates the need for an adaptive solving procedure.

Consider the problem that consists in seeking $\underline{p} : \Omega \to \mathbb{R}$ such that:

$$\begin{cases} -\nabla \cdot (\underline{\mathbf{K}} \nabla \underline{p}) = \underline{f} & \text{in } \Omega \\ \underline{p} = 0 & \text{on } \partial\Omega \end{cases} \tag{5.1}$$

where $\underline{f} : \Omega \to \mathbb{R}$ is a source term in $L^2(\Omega)$, and $\underline{\mathbf{K}} = \kappa \mathbf{I}$ is an uniformly bounded and positive definite permeability tensor. For the sake of simplicity we assume that $\underline{f}$ and κ are cellwise constant with respect to the mesh $\mathcal{T}_h$, and denote by f_i and κ_i their values on a mesh element K_i. The existence of a solution and the convergence of the scheme were proven in [56]. We refer the reader to the latter book for a more rigorous framework of the discretization by finite volume method.

By integrating the law (5.1) over a mesh element K_i, and applying the Stokes formula we obtain:

$$\int_{\partial K_i} (\underline{\mathbf{K}} \nabla \underline{p}) \cdot \boldsymbol{n}_i ds + \int_{K_i} \underline{f} dx = \sum_{\sigma \in \mathcal{F}_i} \int_{\sigma} (\underline{\mathbf{K}} \nabla \underline{p}) \cdot \boldsymbol{n}_{\sigma,i} d\sigma + \int_{K_i} \underline{f} dx = 0; \tag{5.2}$$

If we introduce the discrete unknowns p_i per mesh element K_i and define a cellwise constant function $\underline{p}_h$ that takes the values p_i on the mesh elements K_i, then the fluxes $\int_{\sigma} (\underline{\mathbf{K}} \nabla \underline{p}) \cdot \boldsymbol{n}_{\sigma,i} d\sigma$ can be approximated as functions of the discrete unknowns:

$$\int_{\sigma} (\underline{\mathbf{K}} \nabla \underline{p}) \cdot \boldsymbol{n}_{\sigma,i} d\sigma \approx F_{i,\sigma}(\underline{p}_h);$$

where we have in a finite volume scheme with two point flux approximation for example:

$$F_{i,\sigma}(\underline{p}_h) = \begin{cases} T_{\sigma_{i,j}}(p_i - p_j) \quad \text{with} \quad T_{\sigma_{i,j}} = |\sigma| \left(\dfrac{\kappa_i}{d_{i,\sigma}} + \dfrac{\kappa_j}{d_{j,\sigma}} \right)^{-1} \left(\dfrac{\kappa_i}{d_{i,\sigma}} \times \dfrac{\kappa_j}{d_{j,\sigma}} \right) \quad \text{if} \quad \sigma = \sigma_{ij} \not\subset \partial\Omega; \\ T_{\sigma_i}(p_i) \quad \text{with} \quad T_{\sigma_i} = \dfrac{|\sigma| \kappa_i}{d_{i,\sigma}} \quad \text{otherwise.} \end{cases} \tag{5.3}$$

In (5.3), $F_{i,\sigma}(\underline{p}_h)$ denotes a flux through the edge σ, whereas $T_{\sigma_{i,j}}$ and T_{σ_i} are transmissivities. Therefore, (5.2) becomes:

$$\begin{aligned} &\sum_{\sigma_i \in \mathcal{F}_i \cap \mathcal{F}_{\text{int}}} F_{i,\sigma}(\underline{p}_h) + \sum_{\sigma \in \mathcal{F}_i \cap \mathcal{F}_{\text{ext}}} F_{i,\sigma}(\underline{p}_h) = |K_i| f_i \\ &(\sum_{\sigma \in \mathcal{F}_i \cap \mathcal{F}_{\text{ext}}} T_{\sigma_i} + \sum_{\sigma_{i,j} \in \mathcal{F}_i \cap \mathcal{F}_{\text{int}}} T_{\sigma_{i,j}}) p_i - \sum_{\sigma_{i,j} \in \mathcal{F}_i \cap \mathcal{F}_{\text{int}}} T_{\sigma_{i,j}} p_j = |K_i| f_i \end{aligned} \tag{5.4}$$

This means that the vector $\mathbf{x} = (p_i)_{1\leq i\leq n}$ is the solution of the linear system $\mathbf{A}\cdot\mathbf{x} = \mathbf{b}$ where:

$$\forall i,j \in [\![1,n]\!], i\neq j: \quad \mathbf{b}_i := |K_i| f_i \; ; \quad \mathbf{A}_{ii} = \sum_{\sigma_i\in\mathcal{F}_i\cap\mathcal{F}_{\text{ext}}} T_{\sigma_i} + \sum_{\sigma_{i,j}\in\mathcal{F}_i\cap\mathcal{F}_{\text{int}}} T_{\sigma_{i,j}} \; ; \quad \mathbf{A}_{ij} = \begin{cases} T_{\sigma_{i,j}} \text{ if } j\in V(i); \\ 0 \quad \text{otherwise.} \end{cases} \tag{5.5}$$

Remark 5.1. According to (5.3), we notice that $T_{\sigma_{i,j}} = T_{\sigma_{j,i}}$ for arbitrary neighbors i and j. Therefore, it should be noted from the definition (5.5) that the matrix $\mathbf{A}$ is symmetric. In addition, it is diagonally dominant with positive diagonal entries. $\mathbf{A}$ is thus symmetric positive definite.

5.3 Matrix decomposition and local error reduction

An iterative Krylov solver is used to solve the system $\mathbf{A}\cdot\mathbf{x} = \mathbf{b}$. At each iteration j of the iterative solver, we denote by $\mathbf{x}^{(j)}$ the approximate solution obtained, and by $\underline{p}_h^{(j)}$ the associated approximate function.

In Chapter 2, for a finite element discretization we discussed how to derive adaptive preconditioners, based on a posteriori error estimators, in order to effectively decrease the energy norm of the error. The original cellwise constant finite volume approximation $\underline{p}_h \in \mathbb{P}_0(\mathcal{T}_h)$ is not appropriate for energy-norm type a posteriori error estimation as it is only piecewise constant. For this reason, a postprocessed approximation that has more regularity is introduced $\tilde{\underline{p}}_h$ [143, 55].

This time for a finite volume scheme, and as mentioned in the introduction chapter, the a posteriori error analysis allows to derive estimates for the classical associated error norm:

$$\|\underline{K}^{1/2}\nabla(\underline{p} - \tilde{\underline{p}}_h)\|^2_{\mathrm{L}^2(\Omega)} \leq \eta_{\text{disc}} + \eta_{\text{alg}};$$

where η_{disc} and η_{alg} are a posteriori error estimates of the error components: the former term is supposed to approximate the discretization error and the latter the algebraic error [55]. In addition, such quantities can be obtained locally on each mesh element, therefore, analogous to Chapter 2, we use algebraic error estimates at a fixed iteration i of the iterative solver to have a domain decomposition:

$$\begin{aligned} \overline{\Omega}_1 \cup \overline{\Omega}_2 &= \overline{\Omega} \\ \Omega_1^o \cap \Omega_2^o &= \emptyset; \end{aligned}$$

where Ω_1 is the part with the high algebraic error estimates:

$$\sum_{K\in\mathcal{T}_h(\Omega_1)} (\eta^{(i)}_{\text{alg,K}})^2 \gg \sum_{K\in\mathcal{T}_h(\Omega_2)} (\eta^{(i)}_{\text{alg,K}})^2 \tag{5.6}$$

Remark 5.2. From (5.5), we can derive formulas for two matrices $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$ local to Ω_1 and Ω_2

respectively by considering for all $i \neq j$ in $[\![1, n]\!]$ such that $K_i \subset \Omega_1$ and $K_j \subset \Omega_1$:

$$\mathbf{A}^{(1)}_{ii} = \sum_{\sigma_i \in \mathcal{F}_i \cap \mathcal{F}_{\text{ext}}} T_{\sigma_i} + \sum_{\substack{\sigma_{i,l} \in \mathcal{F}_i \cap \mathcal{F}_{\text{int}} \\ l \in V_1(i)}} T_{\sigma_{i,l}} \; ; \quad \mathbf{A}^{(1)}_{ij} = \begin{cases} T_{\sigma_{i,j}} & \text{if} \quad j \in V(i); \\ 0 & \text{otherwise;} \end{cases} \tag{5.7}$$

and for all $i \neq j$ in $[\![1, n]\!]$ such that $K_i \subset \Omega_2$ and $K_j \subset \Omega_2$:

$$\mathbf{A}^{(2)}_{ii} = \sum_{\sigma_i \in \mathcal{F}_i \cap \mathcal{F}_{\text{ext}}} T_{\sigma_i} + \sum_{\substack{\sigma_{i,l} \in \mathcal{F}_i \cap \mathcal{F}_{\text{int}} \\ l \in V_2(i)}} T_{\sigma_{i,l}} \; ; \quad \mathbf{A}^{(2)}_{ij} = \begin{cases} T_{\sigma_{i,j}} & \text{if} \quad j \in V(i); \\ 0 & \text{otherwise;} \end{cases} \tag{5.8}$$

where $V_1(i)$ (resp. $V_2(i)$) denotes the set of indices belonging to $V(i)$ but whose associated polyhedra are included in Ω_1 (resp. Ω_2). Therefore, we can express the global matrix $\mathbf{A}$ as a sum:

$$\mathbf{A} = \mathbf{A}^{(1)} + \mathbf{A}^{(2)} + \mathbf{F}; \tag{5.9}$$

where for all $i \neq j$ in $[\![1, n]\!]$:

$$\mathbf{F}_{ii} = \begin{cases} \displaystyle\sum_{\substack{\sigma_{i,l} \in \mathcal{F}_i \cap \mathcal{F}_{\text{int}} \\ l \in V_2(i)}} T_{\sigma_{i,l}} & \text{if} \quad K_i \in \Omega_1; \\ \displaystyle\sum_{\substack{\sigma_{i,l} \in \mathcal{F}_i \cap \mathcal{F}_{\text{int}} \\ l \in V_1(i)}} & \text{otherwise;} \end{cases} \qquad \mathbf{F}_{ij} = \begin{cases} T_{\sigma_{i,j}} & \text{if} \quad K_i \in \Omega_1 \quad \text{and} \quad j \in V_2(i); \\ T_{\sigma_{i,j}} & \text{if} \quad K_i \in \Omega_2 \quad \text{and} \quad j \in V_1(i); \\ 0 & \text{otherwise;} \end{cases} \tag{5.10}$$

Formula (5.9) recalls the sum splitting of the matrix as detailed in Section 2.4 of Chapter 2 for the finite element discretization method. The difference is that now for the finite volume method, we have a third term $\mathbf{F}$ that is not local to only one subdomain. This is due to the fact that the finite element scheme is by definition vertex-centered whereas the finite volume scheme considered here is cell-centered. This kind of scheme provides an approximation of the solution that is piecewise constant on a primal mesh. That being said, there exist some vertex-centered finite volume schemes that introduce a so-called dual mesh, which is a conforming triangulation of the domain Ω, around the vertices; see, e.g. [53]. However, for the current context of the application, as the a posteriori error estimates implemented in IFPEN's software were derived for cell-centered schemes, we focus on this latter type of schemes in the sequel.

In finite volume method, we cannot reproduce in the language of matrices an inequality that is equivalent to the main starting hypothesis (2.8) as in the case of finite element method (Section 2.2 in Chapter 2) but we have some estimations of the terms involved in that hypothesis. Replacing the exact terms by their estimates in that inequality yields the assumption (5.6).

Accordingly, since we are filtering the mesh elements with high errors in Ω_1, we expect an inequality that can be written in the form:

$$(\mathbf{x}_L - \mathbf{x}_L^{(i)})^{\mathrm{T}} \cdot \mathbf{A}_L \cdot (\mathbf{x}_L - \mathbf{x}_L^{(i)}) \gg (\mathbf{x}_R - \mathbf{x}_R^{(i)})^{\mathrm{T}} \cdot \mathbf{A}_R \cdot (\mathbf{x}_R - \mathbf{x}_R^{(i)}) \tag{5.11}$$

is satisfied where L and R are subsets that include the indices of mesh elements contained in Ω_1 and Ω_2 respectively. We denote by n_L and n_R the sizes of $\mathbf{A}_L$ and $\mathbf{A}_R$ respectively. Their sum is

equal to the size of $\mathbf{A}$: $n_L + n_R = n$.

We recall that since $\mathbf{A}$ is SPD, $\mathbf{A}_L$ and $\mathbf{A}_R$ are SPD as well and the two terms involved in assumption (5.11) are a part of the global energy norm of the initial system:

$$||\mathbf{x} - \mathbf{x}^{(i)}||^2_{\mathbf{A}} = \langle \mathbf{A} \cdot (\mathbf{x} - \mathbf{x}^{(i)}), \mathbf{x} - \mathbf{x}^{(i)} \rangle = \underbrace{\langle \mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(i)}, \mathbf{x} - \mathbf{x}^{(i)} \rangle_L}_{:=L\text{-term}} + \underbrace{\langle \mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(i)}, \mathbf{x} - \mathbf{x}^{(i)} \rangle_R}_{:=R\text{-term}} ,$$

where

$$\begin{aligned} \langle \mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(i)}, \mathbf{x} - \mathbf{x}^{(i)} \rangle_L &= ||(\mathbf{x} - \mathbf{x}^{(i)})_L||^2_{\mathbf{A}_L} + (\mathbf{x} - \mathbf{x}^{(i)})^T_L \cdot \mathbf{A}_{LR} \cdot (\mathbf{x} - \mathbf{x}^{(i)})_R \\ \langle \mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(i)}, \mathbf{x} - \mathbf{x}^{(i)} \rangle_R &= ||(\mathbf{x} - \mathbf{x}^{(i)})_R||^2_{\mathbf{A}_R} + (\mathbf{x} - \mathbf{x}^{(i)})^T_R \cdot \mathbf{A}_{RL} \cdot (\mathbf{x} - \mathbf{x}^{(i)})_L . \end{aligned} \tag{5.12}$$

Due to the symmetry, the coupling terms are identical:

$$(\mathbf{x} - \mathbf{x}^{(i)})^T_R \cdot \mathbf{A}_{RL} \cdot (\mathbf{x} - \mathbf{x}^{(i)})_L = (\mathbf{x} - \mathbf{x}^{(i)})^T_L \cdot \mathbf{A}_{LR} \cdot (\mathbf{x} - \mathbf{x}^{(i)})_R .$$

Assumption (5.11) implies that the $\mathbf{A}_L$-inner product of the error is dominant, and so will be the L-term according to (5.12). Therefore, reducing them may efficiently reduce the energy norm of the error. As the vectors $(\mathbf{x} - \mathbf{x}^{(i)})_L$ and $\mathbf{A}_L \cdot (\mathbf{x} - \mathbf{x}^{(i)})_L$ cannot be computed, we favor the alternative of reducing the partial residual $(\mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(i)})_L$ to decrease the L-term. A straightforward manner to bring that partial residual down to zero is by using a Schur complement reduction that we detail in the next section.

5.4 Adaptive linear solver

In this section, we explain an approach for reducing the dominant part of the algebraic error. This procedure is based on the exact decomposition on the L-block and a Schur complement on the R-block. In the field of linear algebra and the theory of matrices, the literature is rich on these two techniques. For a detailed description of these concepts, we refer the reader to [147, 137] and the references therein. Another popular use for those techniques is the construction of preconditioners. Approximate or inexact factorization is often used as a preconditioner for iterative solvers such as PCG [123]. In domain decomposition methods as well, many research works were made to devise preconditioners for the global matrix $\mathbf{A}$ from techniques that approximately solve the reduced Schur complement system [47, 96]. We also specify that, when a PCG solver is used, the solve procedure described below, which is based on a reduced Schur complement system, is equivalent to iteratively solving the global system $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ with the initial guess and the preconditioner introduced in Chapter 2. Furthermore, we check if the procedure applied in Chapter 2 can be valid for other solvers than PCG.

5.4.1 Schur complement procedure

Let us define the following matrices:

$$\mathbf{S} = \mathbf{A}_R - \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR}; \qquad \mathbf{A}_L = \mathbf{D}_1\mathbf{D}_2 \tag{5.13}$$

$$\mathbf{N}_1 = \mathbf{A}_{RL}\mathbf{D}_2^{-1}; \qquad \mathbf{N}_2 = \mathbf{D}_1^{-1}\mathbf{A}_{LR} \tag{5.14}$$

$$\mathbf{E}_1 = \left[\begin{array}{c|c} \mathbf{D}_1 & 0 \\ \hline \mathbf{N}_1 & \mathbf{I} \end{array}\right]; \qquad \mathbf{E}_2 = \left[\begin{array}{c|c} \mathbf{D}_2 & \mathbf{N}_2 \\ \hline \mathbf{0} & \mathbf{S} \end{array}\right] \tag{5.15}$$

where $\mathbf{I}$ is the identity matrix of the same size as $\mathbf{A}_R$ and $\mathbf{S}$ is referred to as the Schur complement of the R block $\mathbf{A}_R$. Since $\mathbf{A}_L$ is nonsingular (because it is a principal submatrix of $\mathbf{A}$ which is SPD), the Schur complement is well defined. The matrices $\mathbf{D}_1$ and $\mathbf{D}_2$ are exact factors of $\mathbf{A}_L$ where $\mathbf{D}_1$ is the lower triangular factor, and $\mathbf{D}_2$ is the upper triangular one.

With the definitions of (5.14) and (5.15), we have the equality:

$$\mathbf{A} = \mathbf{E}_1\mathbf{E}_2 \tag{5.16}$$

and we can write:

$$\mathbf{A}\cdot\mathbf{x} = \mathbf{b} \Leftrightarrow \begin{cases} \mathbf{E}_1\cdot\mathbf{z} = \mathbf{b}, & \text{(P.1)} \\ \mathbf{E}_2\cdot\mathbf{x} = \mathbf{z}. & \text{(P.2)} \end{cases}$$

Splitting the system $\mathbf{A}\cdot\mathbf{x} = \mathbf{b}$ into two systems related to L and R domains with the Schur complement is tantamount to solve the system (P') in the following two stages:

① Solve of (P.1): The matrix $\mathbf{E}_1$ is lower triangular, and therefore (P.1) represents a forward substitution:

$$\text{(P.1)} \Leftrightarrow \begin{cases} \mathbf{D}_1\cdot\mathbf{z}_L & = \mathbf{b}_L \\ \mathbf{N}_1\cdot\mathbf{z}_L + \mathbf{z}_R & = \mathbf{b}_R \end{cases} \Leftrightarrow \begin{cases} \mathbf{z}_L & = \mathbf{D}_1^{-1}\cdot\mathbf{b}_L \\ \mathbf{z}_R & = \mathbf{b}_R - \mathbf{N}_1\cdot\mathbf{z}_L \end{cases} \tag{5.18}$$

② Solve of (P.2): We first solve a reduced system with the Schur complement $\mathbf{S}$. Then, we perform a backward substitution with an upper triangular matrix $\mathbf{D}_2$.

$$\text{(P.2)} \Leftrightarrow \begin{cases} \mathbf{S}\cdot\mathbf{x}_R & = \mathbf{z}_R \\ \mathbf{D}_2\cdot\mathbf{x}_L + \mathbf{N}_2\cdot\mathbf{x}_R & = \mathbf{z}_L \end{cases} \Leftrightarrow \begin{cases} \mathbf{S}\cdot\mathbf{x}_R = \mathbf{z}_R & \text{(5.19a)} \\ \mathbf{x}_L = \mathbf{D}_2^{-1}\cdot(\mathbf{z}_L - \mathbf{N}_2\cdot\mathbf{x}_R) & \text{(5.19b)} \end{cases}$$

The size of system (5.19a) is smaller than the size of system $\mathbf{A}\cdot\mathbf{x} = \mathbf{b}$ because its size is equal to the size of R domain, where no significant algebraic errors were observed. In order to avoid forming the (possibly dense) matrix $\mathbf{S}$, an iterative Krylov solver can be used for solving (5.19a) as well. Let $\mathbf{x}_R^{(i)}$ be the approximate solution obtained after i iterations. The associated solution $\mathbf{x}_L^{(i)}$ may be calculated by (5.19b) as:

$$\mathbf{x}_L^{(i)} = \mathbf{D}_2^{-1}\cdot(\mathbf{z}_L - \mathbf{N}_2\cdot\mathbf{x}_R^{(i)}) \tag{5.20}$$

The adaptive solving procedure can be summarized in two major stages presented below. Algorithm 1m for the setting up of the method (splitting and permuting) and Algorithm 1n for the computation phase.

Algorithm 1m Error-Based Domain Decomposition

Inputs: Ω

1: Evaluate the local algebraic error over the mesh elements
   $\hookrightarrow \eta_l^{(alg)}$
2: Mark the elements K_l with largest algebraic errors $\eta_l^{(alg)}$
   $\hookrightarrow \Omega_1$-subdomain
3: Extract the node indices associated to elements of Ω_1
   $\hookrightarrow L$-subset
4: Find the complementary of L in the set of node indices
5: Permute the system to obtain a L/R block splitting

Outputs: Ω_1, L, R

Algorithm 1n Schur complement procedure

Inputs: L, R, $\mathbf{A}$, $\mathbf{b}$, $\mathbf{M}_S$

1: Perform an exact factorization on the L-block
2: Solve (P.1) by simple substitution
3: Solve the Schur complement system (5.19a) via an iterative solver with preconditioner $\mathbf{M}_S$
   $\hookrightarrow \mathbf{x}_R$ solution
4: Inject the obtained vector in (5.19b) and proceed by backward and forward substitution
   $\hookrightarrow$ updated $\mathbf{x}_L$ solution

Outputs: $\mathbf{x}_L$, $\mathbf{x}_R$

Remark 5.3. In the first step of Algorithm 1m, the algebraic error estimation can be done in different ways, we cite for example [55, Section 4.] where the authors present a simple and practical way to estimate the algebraic error.

5.4.2 Error reduction properties of the adaptive procedure

We recall in the following lemma some formulas for the residual and the energy norm of the error that hold with the Schur complement method.

Lemma 5.1. *By applying the Schur complement procedure described in Subsection 5.4.1, we obtain at each iteration i of an iterative solver for any $\mathbf{x}_R^{(i)}$ approximating $\mathbf{x}_R$ and associated $\mathbf{x}_L^{(i)}$ given by (5.20):*

$$\begin{cases} \mathbf{b} - \mathbf{A}\cdot\mathbf{x}^{(i)} = \begin{bmatrix} \mathbf{0} \\ \mathbf{z}_R - \mathbf{S}\cdot\mathbf{x}_R^{(i)} \end{bmatrix} & (5.21) \\ ||\mathbf{x}-\mathbf{x}^{(i)}||_{\mathbf{A}}^2 = \langle (\mathbf{x}-\mathbf{x}^{(i)})_{/R}, \mathbf{z}_R - \mathbf{S}\cdot\mathbf{x}_R^{(i)} \rangle & (5.22) \end{cases}$$

Proof.

$$\begin{aligned} (\mathbf{b}-\mathbf{A}\cdot\mathbf{x}^{(i)})_{/L} &\overset{(2.24)}{=} \mathbf{A}_L\cdot(\mathbf{x}-\mathbf{x}^{(i)})_{/L} + \mathbf{A}_{LR}\cdot(\mathbf{x}-\mathbf{x}^{(i)})_{/R} \overset{(2.23)}{=} \mathbf{b}_L - \mathbf{A}_L\cdot\mathbf{x}_L^{(i)} - \mathbf{A}_{LR}\cdot\mathbf{x}_R^{(i)} \\ &\overset{(5.20)}{=} \mathbf{b}_L - (\mathbf{b}_L - \mathbf{D}_1\mathbf{N}_2\cdot\mathbf{x}_R^{(i)}) - \mathbf{A}_{LR}\cdot\mathbf{x}_R^{(i)} \overset{(5.14)}{=} \mathbf{0} \end{aligned}$$

And similarly,

$$\begin{aligned} (\mathbf{b}-\mathbf{A}\cdot\mathbf{x}^{(i)})_{/R} &\overset{(2.24)}{=} \mathbf{A}_{RL}\cdot(\mathbf{x}-\mathbf{x}^{(i)})_{/L} + \mathbf{A}_R\cdot(\mathbf{x}-\mathbf{x}^{(i)})_{/R} \overset{(2.23)}{=} \mathbf{b}_R - \mathbf{A}_{RL}\cdot\mathbf{x}_L^{(i)} - \mathbf{A}_R\cdot\mathbf{x}_R^{(i)} \\ &\overset{(5.14)}{=} \mathbf{b}_R - \mathbf{N}_1\mathbf{D}_2\cdot\mathbf{x}_L^{(i)} - \mathbf{A}_R\cdot\mathbf{x}_R^{(i)} \overset{(5.20)}{=} \mathbf{b}_R + \mathbf{N}_1\cdot(\mathbf{N}_2\cdot\mathbf{x}_R^{(i)} - \mathbf{z}_L) - \mathbf{A}_R\cdot\mathbf{x}_R^{(i)} \\ &\overset{(5.18)\ \&\ (5.13)}{=} \mathbf{z}_R - \mathbf{S}\cdot\mathbf{x}_R^{(i)} \end{aligned}$$

□

By looking at the block expression of the residual vector in (5.21), it can be stated that the L-part of that vector is eliminated and there remains only the R-part. Consequently, in Formula (5.22) we got rid of the L-term which was dominant and that the energy norm now depends only on the scalar product between $(\mathbf{x} - \mathbf{x}^{(i)})_{/R}$ the error vector projected on R, and the residual of the solution of (5.19a).
We would add that in practice, the only components we actually need to compute, for the whole solving process, are the factors $\mathbf{D}_1$ and $\mathbf{D}_2$. The benefit of solving (5.19a) by some Krylov method is that it only requires matrix-vector products $\mathbf{S}$ times a vector $\mathbf{w}$ which can be performed without computing the entries of the Schur complement matrix $\mathbf{S}$. This way, the solving process combines a direct solver in the subdomain L with an iterative solver for the subdomain R. Thus, it can be seen as an hybrid direct/iterative solver. As for the choice of the preconditioning to be used during the iterative solve, we refer to [126, 32, 97] for a range of preconditioners for the Schur complement. Sometimes, a preconditioner $\mathbf{M}_R$ of the submatrix $\mathbf{A}_R$ can be used to precondition the Schur complement $\mathbf{S}$.

So far, we have shown that decreasing the algebraic error by reducing the residual on the L-part to zero is achievable for other solvers than PCG. This represents an extension of the results of Chapter 2.

5.5 Numerical results

In this section, we present numerical results of tests where we apply the adaptive solve procedure in a reservoir simulation for heterogeneous porous media. The model problem is a single phase flow model, for which we consider two types of problems: steady (as introduced in Section 5.2) and unsteady (see Section 5.5.2.1). We first validate the procedure in the simpler steady case with an initial prototype. Once the method validated, we assess its performance on the real simulator with the unsteady case. The simulations are run on IFPEN's prototype for reservoir simulation.

In the sequel, we solve the linear systems stemming from the PDEs by the adaptive Schur procedure and compare it to the classical solve procedure. Both procedures employ the same Krylov solver and the same incomplete factorization. The classical solve is performed with an incomplete factorization preconditioner for the global matrix $\mathbf{A}$, whereas the adaptive Schur procedure uses an incomplete factorization of the submatrix $\mathbf{A}_R$ to precondition the reduced Schur complement system. We also consider a stopping threshold value of 10^{-6} for the euclidean norm of the residual.

5.5.1 Steady problem: Heterogeneous media and uniform mesh refinement

For the first test case, we deal with a steady problem. As it generates one single system, we extract data (matrix, right hand side vector and error estimates) from the simulator and solve the linear system with a prototype of the adaptive approach that uses the no-fill Incomplete Cholesky factorization and a PCG solver.

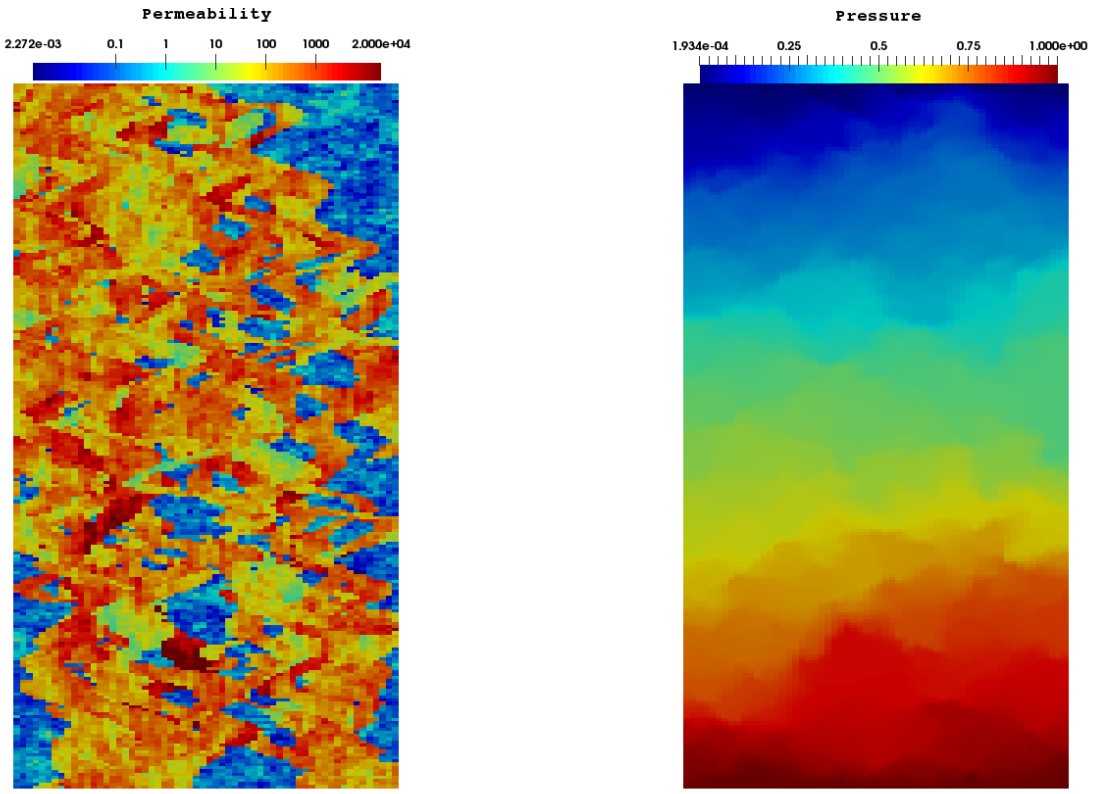


Figure 5.1 – *SPE10 permeability (left) and pressure field (right). Section 5.5.1.*

In this test, we use the same configuration described in [102, Section 6.3]. We consider a heterogeneous porous media with domain $(0, 1200) \times (0, 2200)$ partitioned by a grid of 60×220 rectangular cells. The permeability tensor corresponds to that of the layer 85 of the tenth SPE comparative solution project model field [38]. Figure 5.1 shows the permeability field (on the left) and the pressure field (on the right). The source term is $f = 0$. We have tested four values (0.10, 0.15, 0.25 and 0.33) for the size percentage $\delta := \dfrac{n_L}{n}$ (see Section 5.3). The evolution of the energy norm is displayed in the graph of Figure 5.2.

We notice from the graph above that by initially taking a subset L which is ten times smaller than the size of the global system, we get a decrease of almost 30% in the number of iterations for the adaptive solve with respect to the standard one. The convergence is still accelerated by a wider coverage of the high error areas. This is reflected in the decrease of the number of iterations when we progressively increase the parameter δ, until we obtain a speedup of more than 50% with respect to the standard solve for $\delta = 0.33$.

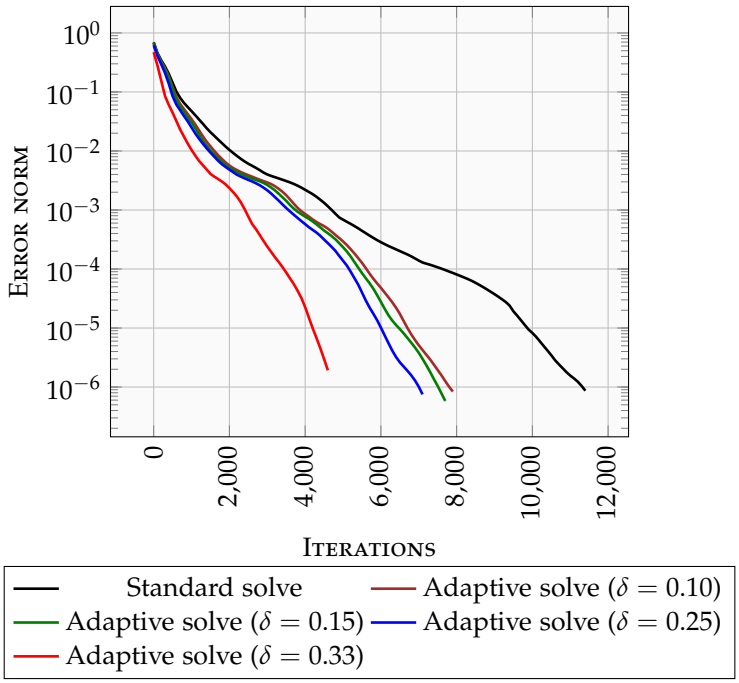


Figure 5.2 – *Evolution of the energy norm of the error with the standard and adaptive solve procedures (steady case).*

5.5.2 Unsteady problem: Heterogeneous media and uniform mesh refinement

For the second test case, we handle an unsteady problem of a single phase flow model.

5.5.2.1 The model

Often used in reservoir simulation, this unsteady model describes the flow of a single fluid through a porous medium $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$, over a certain time interval. On the one hand, we consider the characteristics of the fluid that follow. We denote by $\underline{p}$ the pressure of the fluid, by ρ its mass density, by μ its viscosity, by c_f its compressibility and by $\mathbf{v}$ the fluid velocity. On the other hand, the physical characteristics of the porous medium are its porosity ϕ, and its absolute permeability tensor $\underline{\mathbf{K}}$. This latter measures the ability of the porous medium to transmit fluid in each direction. We also denote by c_R the rock compressibility and by ρ^0 the fluid density at a reference pressure $\underline{p}^0$.

It is assumed that the mass diffusion and mass dispersion fluxes are negligible and that no fluid mass can cross the fluid-solid interface. Then, the conservation of mass is expressed by the following equation:

$$\frac{\partial(\phi\rho)}{\partial t} = -\nabla \cdot (\rho \mathbf{v}) + q, \tag{5.23}$$

where q is the source or sink term that is square integrable in time and space.

In addition, Darcy's law gives the expression of the fluid velocity:

$$\mathbf{v} = -\frac{1}{\mu}\underline{\mathbf{K}}(\nabla \underline{p} - \rho g \nabla z); \tag{5.24}$$

where g is the magnitude of the gravitational acceleration and z is the depth.

An equation of state gives the relationship between the fluid compressibility and the partial derivative of the density with respect to the pressure evaluated at a fixed temperature T:

$$c_f = \frac{1}{\rho}\left.\frac{\partial \rho}{\partial \underline{p}}\right|_T. \tag{5.25}$$

Similarly, the rock compressibility is defined by

$$c_R = \frac{1}{\phi}\frac{d\phi}{d\underline{p}}. \tag{5.26}$$

The time differentiation of $\phi\rho$ in (5.23) yields:

$$\left(\phi\frac{\partial \rho}{\partial \underline{p}} + \rho\frac{d\phi}{d\underline{p}}\right)\frac{\partial \underline{p}}{\partial t} = -\nabla\cdot(\rho\mathbf{v}) + q.$$

By injecting the compressibility formulae (5.25), (5.26) and the momentum conservation's law (5.24), in the mass conservation's law (5.23), we obtain

$$\rho(c_f + c_R)\phi\frac{\partial \underline{p}}{\partial t} = \nabla\cdot\left(\frac{\rho}{\mu}\underline{\mathbf{K}}(\nabla \underline{p} - \rho g \nabla z)\right) + q. \tag{5.27}$$

We consider that the medium contains a single fluid (oil or gas) that is slightly compressible. Thus, the fluid compressibility stays constant when the pressure varies within a certain range of values. In this case, integrating (5.25) yields:

$$\rho = \rho^0 e^{c_f(\underline{p} - \underline{p}^0)}. \tag{5.28}$$

Hence, with (5.28), the governing PDE (5.27) is a parabolic equation for the main unknown which is the pressure $\underline{p}$. For proofs of the existence, uniqueness and regularity of a solution to this system, and for discretization and linearization approaches, we refer to [33] and references therein.

5.5.2.2 Simulation results

As far as the computing framework is concerned, we have implemented GMRES solver and the adaptive solve procedure on MCGSolver [7, 8]. For exact and inexact LU factorizations, we used the library Eigen [72]. The transfer of error estimates between the linear solver (MCGSolver) and the simulator is operated by the ALIEN interface (see Section 1.1.4 in Chapter 1). Note that, as we consider a horizontal 2D case, gravitational effects are not taken into account in the numerical tests. A simulation over a 24-hour period was conducted in this study. We consider a 2D cartesian grid (60×220). At each time t_i, an initial time step $\Delta t_i^{(0)}$ is set, a linear system

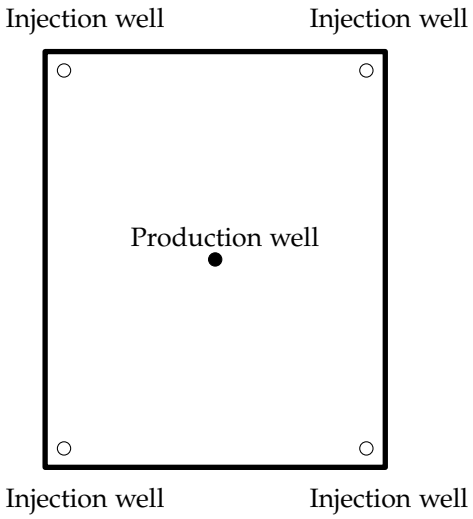


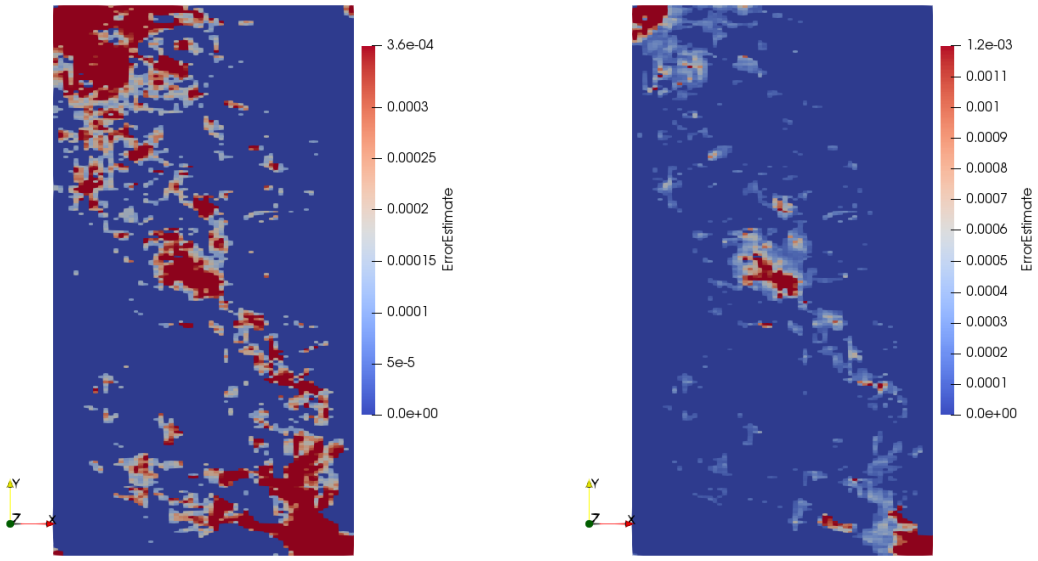
Figure 5.3 – *Configuration for the numerical test case of Section 5.5.2*

is generated and solved with GMRES solver. Note that usually the non-linear solver does not converge when the linear system's solve did not converge. Note also that when the non-linear solver does not converge, the time step is halved $\Delta t_i^{(j+1)} := \dfrac{\Delta t_i^{(j)}}{2}$ until reaching convergence with $j = j_c$. In this case, $\left(\Delta t_i^{(j)}\right)_{j<j_c}$ (resp. $\Delta t_i^{(j_c)}$) are called failed (resp. accepted) time steps at the time t_i. The next time is set from the accepted time step $t_{i+1} := t_i + \Delta t_i^{(j_c)}$.

The size of the system matrix $\mathbf{A}$ is $12\,997 \times 12\,997$, whereas the size of the submatrix $\mathbf{A}_L$ is 417×417. For the sake of comparison, the solves in the simulation are carried out according to two procedures. The first is the standard solve using an ILU(0) factorization of the matrix $\mathbf{A}$ as a preconditioner. The second one is the adaptive procedure described in Section 5.4.1.

The locations of the wells at the reservoir are indicated on Figure 5.3 above. The wells are arranged in a five-spot pattern, with the production well positioned at the center. The distribution of a posteriori error estimates during the beginning and the end of the simulation is plotted on Figure 5.4.

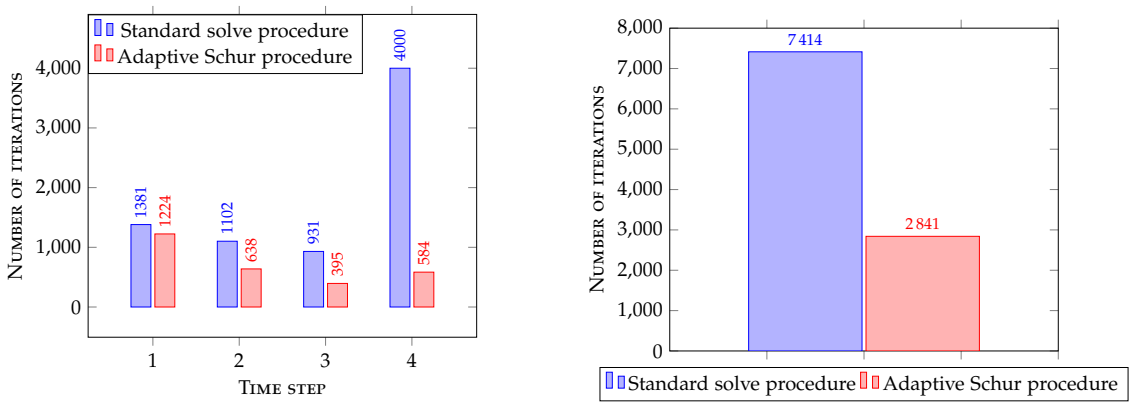
We are interested in the number of iterations needed at every time step of the simulation for the standard solve procedure and the adaptive Schur procedure respectively. Whenever that number reaches 4 000, which is the maximum number of iterations allowed, this indicates a failed time step. One can observe that the first failed time step occurs at the sixth time t_6 for the standard procedure. Therefore, the subsequent times $\left(t_j\right)_{j>6}$ differ from the standard and adaptive procedure. Yet, we still can compare the two procedures during the first time steps that are identical and common for them both. The number of iterations for the times in question is displayed in Figure 5.5. The evolution of the norm of the residual over iterations with both procedures is plotted for each time step in Figure 5.6. With respect to the standard solve procedure, we observe a speedup with the adaptive Schur procedure in the first common time steps.



(a) A posteriori error estimates at the beginning of the simulation

(b) A posteriori error estimates at the end of the simulation

Figure 5.4 – *Distribution of a posteriori error estimates during the single phase flow simulation*

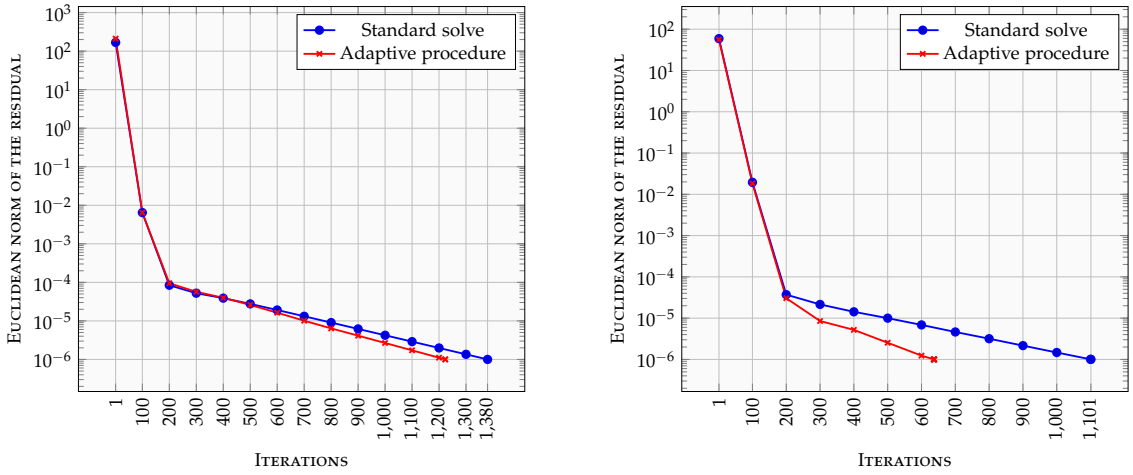


(a) Per time step

(b) Total

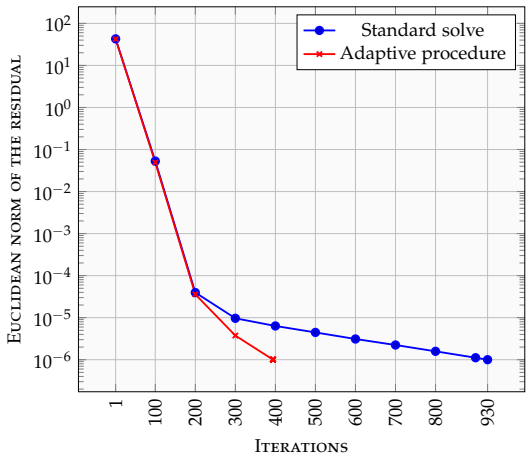
Figure 5.5 – *Number of iterations during the first four time steps after initialization*

Moreover, this trend is confirmed on a larger scale for a whole simulation. The total numbers of time steps and GMRES iterations are reported in Figure 5.7. The charts of this latter indeed indicate that more time steps and iterations are needed for the simulation when the standard solve procedure is employed compared to the adaptive one.

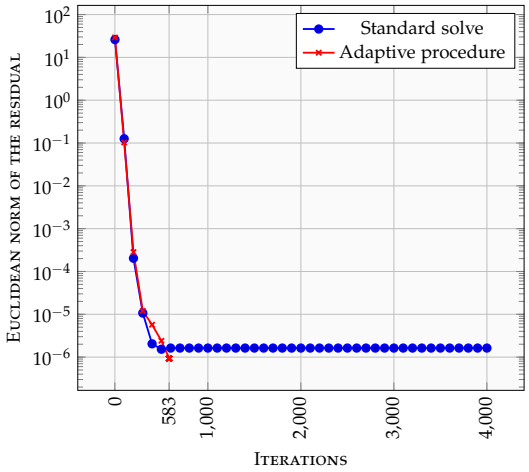


(a) Time step n 1

(b) Time step n 2

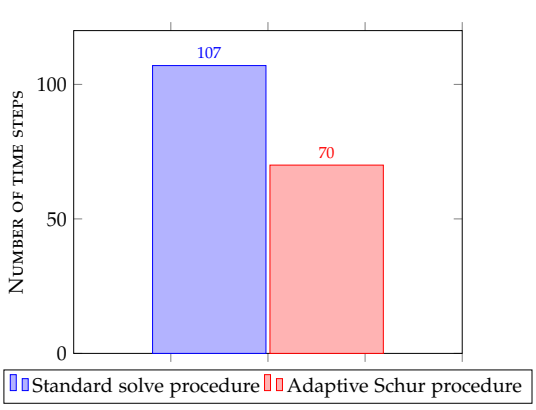


(c) Time step n 3

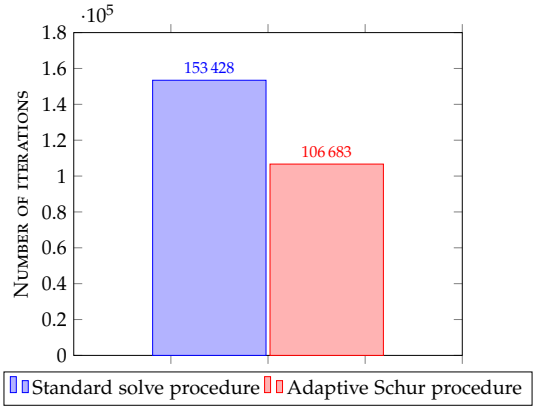


(d) Time step n 4

Figure 5.6 – *Convergence of the solve procedures during the first five time steps after initialization.*



(a) Time steps



(b) Iterations

Figure 5.7 – *The total number of time steps and iterations needed for the whole simulation with the standard and adaptive procedures. The maximum number of iterations allowed per solve is set to* 4 000.

The solve times are collected in Table 5.1. We notice that the standard solve takes less time than the adaptive procedure in the first two time steps, and more time on the third and fourth time steps.

Table 5.1 – *The solve times (in seconds) for the first four time steps of the simulation with the standard and adaptive procedures.*

	Standard Solve	**Adaptive Schur**
Time step n 1	7.15	14.26
Time step n 2	5.59	6.91
Time step n 3	4.68	3.84
Time step n 4	21.06	5.84

5.6 Conclusion

In this chapter, we have adapted the a posteriori error estimates hypothesis for the finite volume discretization of the model problem. Then we presented the adaptive Schur procedure to exploit this hypothesis in order to effectively reduce the algebraic error and accelerate the convergence. Lastly, we showed the numerical results of the method applied to the framework of reservoir simulation. The results of the initial tests are encouraging. The comparison with the standard solve illustrates the performance of the adaptive procedure and in particular reveals that a significant speedup in terms of the number of time steps and iterations can be achieved. Yet, the results for the time gain of this method are not as conclusive as for the number of iterations and time steps because the implementation of the approach was unsophisticated making the computations with the Schur complement costly in time. There is certainly room for improvement in this regard. For future perspectives, we could rely on hierarchical matrices [73] and the relevant techniques that are efficient for data-sparse representations of certain densely populated matrices such as the Schur complement in the elliptic models to reduce the costs in floating point operations (the complexity) and time. In addition, further test cases are envisaged, such as two-phase or multiphase flow models.

Conclusion

Choosing the appropriate preconditioning method for iterative algebraic solvers in numerical solution of partial differential equations is a broad subject of research. The fact that the linear systems are various and the characteristics of the matrices handled differ according not only to the type of the application but also to the test case makes choosing the convenient preconditioner a real challenge for numerical engineers and mathematicians. In this thesis we have investigated this issue, and devised some strategies that allow to monitor the choice of the preconditioner according to the information stemming from a posteriori error estimates.

The first point of the thesis is that error estimates can provide a reliable information about the global error norms and thus gives an idea of each system's complexity during the simulation. On the basis of such information, we have been able to choose the appropriate preconditioner for the iterative solver at every time step of the simulation. From that perspective, error estimates may be considered as an intelligent decision aid tool as far as the preconditioning strategy is concerned (first part of Chapter 1).

The second point concerns the usefulness of error estimates when used locally. Within a domain decomposition framework, they help identify the areas of the domain where the error is significant. We have experimentally investigated the existence of a link between the local algebraic error and the conditioning of the corresponding diagonal blocks of the matrix. We have noticed that in general when the error is high, the corresponding diagonal submatrix is ill-conditioned. As a consequence, we suggested a block-diagonal preconditioner where the robustness of the blocks was adjusted to the magnitude of local error estimates of the corresponding subdomains. In terms of performance, this solution offers a compromise between the number of iterations and the complexity of building the preconditioner. Yet, we do not think a naive block-diagonal preconditioner is sufficient to significantly improve the convergence and that a more sophisticated preconditioning strategy is needed instead (second part of Chapter 1).

The third point concerns the adaptive preconditioner that is built with an exact factorization of the block where the local error estimates are high. When used with PCG in combination with a specific initial guess, we have noticed that it enables to efficiently reduce the algebraic error and

enhance the solve convergence. However, this approach has its limits as regards the inversion of the block with high error. It also seems to have difficulties when dealing with much scattered high error zones (Chapter 2).

To circumvent the computational burden of exact matrix inversion, approximate adaptive preconditioners are introduced as alternatives to the original ones. With respect to the latter, the former use approximate inverses. The efficiency of this type of preconditioner depends on the quality of the approximation considered. In the case of a low rank approximation, the numerical results show that the approximate adaptive preconditioner yields an error reduction that is very close to the exact adaptive preconditioner even by computing a few eigenvalues only (Chapter 3).

To control the growth rate of a relevant seminorm of the error with a fixed-point iteration, a second variant of the adaptive preconditioner is needed. With this preconditioner, we are able to prove that the growth rate of the error seminorm in question depends on the largest eigenvalue of a specific block of the preconditioned operator $\mathbf{M}^{-1}\mathbf{A}$. Even if the theory stands for fixed-point iteration scheme, the experimentation shows that this kind of preconditioner behaves well with PCG solver also, and can outperform the first variant (Chapter 4).

Finally, the last part of this study recalls a solve procedure, based on a Schur complement, that is equivalent, in the case of a PCG, to solve the initial system with the first adaptive preconditioner introduced. This part investigates the performance of this procedure when applied to the framework of reservoir simulation. Even though significant speedups can be obtained with the adaptive solve procedure, there remains sufficient scope for further improvements as concerns the Schur complement computations, and extension of the theory and experiments to other PDE models (Chapter 5).

Many questions regarding the enhancement of solving linear systems, stemming from PDEs, using error estimates remain open. We list some perspectives related to the topics in this thesis that we would like to explore in the future:

— the use of hierarchical matrices for the Schur complement part,

— efficient implementation on multicore GPU-accelerated architectures.

Bibliography

[1] *How to use EISPACK*, pages 5–76. Springer Berlin Heidelberg, Berlin, Heidelberg, 1977.

[2] Y. Achdou and F. Nataf. Low frequency tangential filtering decomposition. *Numerical Linear Algebra with Applications*, 14(2):129–147.

[3] M. Ainsworth. A synthesis of a posteriori error estimation techniques for conforming, non-conforming and discontinuous Galerkin finite element methods. In *Recent advances in adaptive computation*, volume 383 of *Contemp. Math.*, pages 1–14, Providence, RI, 2005. Amer. Math. Soc.

[4] M. Ainsworth and J. T. Oden. A posteriori error estimation in finite element analysis. *Comput. Methods Appl. Mech. Engrg.*, 142(1-2):1–88, 1997.

[5] G. Allaire and S. M. Kaber. *Numerical Linear Algebra*. Springer New York, 2008.

[6] G. Alleon, M. Benzi, and L. Giraud. Sparse approximate inverse preconditioning for dense linear systems arising in computational electromagnetics. *Numer. Algorithms 16, 1*, 1997.

[7] A. Anciaux-Sedrakian, J. Eaton, J.-M. Gratien, T. Guignon, P. Havé, C. Preux, and O. Ricois. Will GPGPUs be finally a credible solution for industrial reservoir simulators? In *SPE Reservoir Simulation Symposium*, volume 1. Society of Petroleum Engineers, 2015.

[8] A. Anciaux-Sedrakian, P. Gottschling, J.-M. Gratien, and T. Guignon. Survey on efficient linear solvers for porous media flow models on recent hardware architectures. *Oil Gas Sci. Technol. - Rev. IFP Energies nouvelles*, 69(4):753–766, 2014.

[9] M. Arioli, E. H. Georgoulis, and D. Loghin. Stopping criteria for adaptive finite element solvers. *SIAM J. Sci. Comput.*, 35(3):A1537–A1559, 2013.

[10] M. Arioli, J. Liesen, A. Miedlar, and Z. Strakoš. Interplay between discretization and algebraic computation in adaptive numerical solutionof elliptic pde problems. *GAMM-Mitteilungen*, 36(1):102–129, 2013.

[11] M. Arioli, D. Loghin, and A. J. Wathen. Stopping criteria for iterations in finite element methods. *Numer. Math.*, 99(3):381–410, 2005.

[12] O. Axelsson. *Iterative Solution Methods*. Cambridge University Press, New York, 1994.

[13] O. Axelsson and I. Kaporin. Optimizing two-level preconditionings for the conjugate gradient method. In Svetozar Margenov, Jerzy Waśniewski, and Plamen Yalamov, editors, *Large-Scale Scientific Computing*, pages 3–21, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[14] I. Babuška and W. C. Rheinboldt. Error estimates for adaptive finite element computations. *SIAM J. Numer. Anal.*, 15(4):736–754, 1978.

[15] D. Bai and A. Brandt. Local mesh refinement multilevel techniques. *SIAM J. Sci. Statist. Comput.*, 8(2):109–134, 1987.

[16] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. D. Dalcin, V. Eijkhout, W. Gropp, D. Kaushik, et al. Petsc users manual revision 3.8. Technical report, Argonne National Lab.(ANL), Argonne, IL (United States), 2017.

[17] R. E. Bank and A. H. Sherman. An adaptive, multilevel method for elliptic boundary value problems. *Computing*, 26(2):91–105, 1981.

[18] R. E. Bank and R. K. Smith. A posteriori error estimates based on hierarchical bases. *SIAM J. Numer. Anal.*, 30(4):921–935, 1993.

[19] R. Becker, C. Johnson, and R. Rannacher. Adaptive error control for multigrid finite element methods. *Computing*, 55(4):271–288, 1995.

[20] M. W. Benson. Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems. *Util. Math.*, 22:127–140, 1982.

[21] M. Benzi and M. Tůma. A comparative study of sparse approximate inverse preconditioners. *Applied Numerical Mathematics*, 30(2-3):305–340, 1999.

[22] N. Biggs. *Algebraic graph theory*, volume 67. Cambridge university press, 1993.

[23] E. G. Boman and M. M. Wolf. A nested dissection approach to sparse matrix partitioning for parallel computations. 2007.

[24] C. F. Borges and W. B. Gragg. A parallel divide and conquer algorithm for the generalized real symmetric definite tridiagonal eigenproblem. In L. Reichel, A. Ruttan, and R.S. Varga, editors, *Numerical Linear Algebra and Scientific Computing*, pages 10–28, Berlin, Germany, 1993. De Gruyter.

[25] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Math. Comp.*, 31(138):333–390, 1977.

[26] W. L. Briggs, H. Van Emden, and S. F. McCormick. *A Multigrid Tutorial*. Society for Industrial and Applied Mathematics, 2000.

[27] X. C. Cai and M. Sarkis. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM Journal on Scientific Computing*, 21(2):792–797, 1999.

[28] J. J. Camata, A. L. G. A. Coutinho, A. M. P. Valli, L. Catabriga, and G. F. Carey. Block ILU preconditioners for parallel AMR/C simulations. *Proceedings of 30th CILAMCE, Armação dos Búzios, Rio de Janeiro/Brazil*, 2009.

[29] H. Cao, H. A. Tchelepi, J. R. Wallis, and H. E. Yardumian. Parallel scalable unstructured CPR-type linear solver for reservoir simulation. 2005.

[30] C. Carstensen. A posteriori error estimate for the mixed finite element method. *Math. Comp.*, 66(218):465–476, 1997.

[31] M. J. Castro-Díaz, F. Hecht, B. Mohammadi, and O. Pironneau. Anisotropic unstructured mesh adaption for flow simulations. *Internat. J. Numer. Methods Fluids*, 25(4):475–491, 1997.

[32] T. F. Chan and T. P. Mathew. Domain decomposition algorithms. In *Acta Numerica 1994*, pages 61–143. Cambridge University Press, 1994.

[33] Z. Chen, G. Huan, and Y. Ma. *Computational Methods for Multiphase Flows in Porous Media (Computational Science and Engineering 2)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2006.

[34] E. W. Cheney and D. R. Kincaid. *Numerical Mathematics and Computing*. Brooks Cole, 2007.

[35] A.-L. Cholesky. Sur la résolution numérique des systèmes d'équations linéaires. *Bulletin de la Sabix [En ligne]*, 2005.

[36] E. Chow and A. Patel. Fine-grained parallel incomplete lu factorization. *SIAM Journal on Scientific Computing*, 37(2):C169–C193, 2015.

[37] E. Chow and Y. Saad. Approximate inverse preconditioners via sparse-sparse iterations. *SIAM Journal on Scientific Computing*, 19(3):995–1023, 1998.

[38] M. A. Christie and M. J. Blunt. Tenth SPE comparative solution project : A comparison of upscaling techniques. *SPE Reservoir Evaluation & Engineering*, 4(4):308–317, 2001.

[39] E. S. Coakley and V. Rokhlin. A fast divide-and-conquer algorithm for computing the spectra of real symmetric tridiagonal matrices. *Applied and Computational Harmonic Analysis*, 34(3):379 – 414, 2013.

[40] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comput.*, 9(3):251–280, March 1990.

[41] J. D. F. Cosgrove, J. C. Diaz, and A. Griewank. Approximate inverse preconditionings for sparse linear systems. *International journal of computer mathematics*, 44(1-4):91–110, 1992.

[42] J. J. M. Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numerische Mathematik*, 36(2):177–195, Jun 1980.

[43] Y.-H. Dai and Yuan J. Study on semi-conjugate direction methods for non-symmetric systems. *International Journal For Numerical Methods In Engineering*, 60:1383–1399, 2004.

[44] P. Destuynder and B. Métivet. Explicit error bounds in a conforming finite element method. *Math. Comp.*, 68(228):1379–1396, 1999.

[45] D. A. Di Pietro, E. Flauraud, M. Vohralík, and S. Yousef. A posteriori error estimates, stopping criteria, and adaptivity for multiphase compositional Darcy flows in porous media. *J. Comput. Phys.*, 276:163–187, 2014.

[46] D. A. Di Pietro, M. Vohralík, and S. Yousef. Adaptive regularization, linearization, and discretization and a posteriori error control for the two-phase Stefan problem. *Math. Comp.*, 84(291):153–186, 2015.

[47] V. Dolean, P. Jolivet, and F. Nataf. *An Introduction to Domain Decomposition Methods: Algorithms, Theory, and Parallel Implementation*. SIAM, 2015.

[48] X. Dong and G. Cooperman. A bit-compatible parallelization for ilu (k) preconditioning. In *European Conference on Parallel Processing*, pages 66–77. Springer, 2011.

[49] W. Dörfler. A convergent adaptive algorithm for Poisson's equation. *SIAM J. Numer. Anal.*, 33(3):1106–1124, 1996.

[50] V. Eijkhout. Overview of iterative linear system solver packages. Technical report, Department of Computer Science, University of Tennessee, Knoxville, Knoxville, TN 37996, USA, December 1998.

[51] T. Eirola and O. Nevanlinna. Accelerating with rank-one updates. *Linear Algebra and its Applications*, 121:511 – 520, 1989.

[52] M. Embree. The tortoise and the hare restart GMRES. *SIAM Review*, 45:259–266, 2003.

[53] C. Erath and D. Praetorius. Adaptive vertex-centered finite volume methods with convergence rates. *SIAM Journal on Numerical Analysis*, 54(4):2228–2255, 2016.

[54] T. Ericsson and A. Ruhe. The spectral transformation lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems. *Mathematics of Computation*, 35(152):1251–1268, 1980.

[55] A. Ern and M. Vohralík. Adaptive inexact Newton methods with a posteriori stopping criteria for nonlinear diffusion PDEs. *SIAM J. Sci. Comput.*, 35(4):A1761–A1791, 2013.

[56] R. Eymard, T. Gallouët, and R. Herbin. Finite volume methods. *Handbook of numerical analysis*, 7:713–1018, 2000.

[57] P. O. Frederickson. Fast approximate inversion of large sparse linear systems. *Math. Report*, 7, 1975.

[58] P. J. Frey and F. Alauzet. Anisotropic mesh adaptation for CFD computations. *Comput. Methods Appl. Mech. Engrg.*, 194(48):5068–5082, 2005.

[59] F. Gibou and C. Min. On the performance of a simple parallel implementation of the ilu-pcg for the poisson equation on irregular domains. *Journal of Computational Physics*, 231(14):4531 – 4536, 2012.

[60] G. H. Golub. Numerical methods for solving linear least squares problems. *Numer. Math.*, 39:206–216, 1965.

[61] G. H. Golub and Z. Strakoš. Estimates in quadratic formulas. *Numerical Algorithms*, 8(2):241–268, Sep 1994.

[62] G. H. Golub and C. F. Van Loan. Matrix computations, 3rd, 1996.

[63] N. I. M. Gould and J. A. Scott. Sparse approximate-inverse preconditioners using norm-minimization techniques. *SIAM Journal on Scientific Computing*, 19(2):605–625, 1998.

[64] J.-M. Gratien, T. Guignon, J.-F. Magras, P. Quandalle, and O. Ricois. How to improve the scalability of an industrial parallel reservoir simulator. *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems*, pages 114–121, 01 2006.

[65] L. Grigori, S. Cayrols, and J. W. Demmel. Low Rank Approximation of a Sparse Matrix Based on LU Factorization with Column and Row Tournament Pivoting. *SIAM Journal on Scientific Computing*, 40(2):C181–C209, January 2018.

[66] L. Grigori, P. Y. David, J. Demmel, and S. Peyronnet. Brief Announcement: Lower bounds on communication for sparse Cholesky factorization of a model problem. In Friedhelm Meyer auf der Heide and Cynthia A. Phillips, editors, *SPAA 2010: Proceedings of the 22nd Annual ACM Symposium on Parallelism in Algorithms and Architectures, Thira, Santorini, Greece, June 13-15, 2010*, pages 79–81. ACM, 2010.

[67] L. Grigori, F. Nataf, and S. Yousef. Robust algebraic Schur complement preconditioners based on low rank corrections. Research Report RR-8557, INRIA, July 2014.

[68] R. Grimes, J. Lewis, and H. Simon. A shifted block lanczos algorithm for solving sparse symmetric generalized eigenproblems. *SIAM Journal on Matrix Analysis and Applications*, 15(1):228–272, 1994.

[69] W. D. Gropp and D. E. Keyes. Domain Decomposition on Parallel Computers. *Proceedings of the 2nd International Conference on Domain Decomposition Methods*, pages 260–268, 1989.

[70] M. J. Grote and T. Huckle. Parallel Preconditioning with Sparse Approximate Inverses. *SIAM J. of Scient. Comput.*, 18(3):838–853, 1997.

[71] M. Gu and S. Eisenstat. A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem. *SIAM Journal on Matrix Analysis and Applications*, 16(1):172–191, 1995.

[72] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[73] W. Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. part i: Introduction to $\mathcal{H}$-matrices. *Computing*, 62(2):89–108, 1999.

[74] P. Havé, R. Masson, F. Nataf, M. Szydlarski, H. Xiang, and T. Zhao. Algebraic domain decomposition methods for highly heterogeneous problems. *SIAM J. Scientific Computing*, 35, 2013.

[75] B. Hendrickson and T. G. Kolda. Partitioning rectangular and structurally nonsymmetric sparse matrices for parallel computation. *SIAM Journal on Scientific Computing*, 21(6):2048–2072, 2000.

[76] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6), 1952.

[77] N. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, second edition, 2002.

[78] T. Huckle. Approximate sparsity patterns for the inverse of a matrix and preconditioning. *Applied numerical mathematics*, 30(2-3):291–303, 1999.

[79] P. Jiránek, Z. Strakoš, and M. Vohralík. A posteriori error estimates including algebraic error and stopping criteria for iterative solvers. *SIAM J. Sci. Comput.*, 32(3):1567–1590, 2010.

[80] P. Jolivet, V. Dolean, F. Hecht, F. Nataf, C. Prud'homme, and N. Spillane. High performance domain decomposition methods on massively parallel architectures with freefem++. *Journal of Numerical Mathematics*, 20:287–302, 03 2013.

[81] W. A. Joubert. On the convergence behavior of the restarted GMRES algorithm for solving nonsymmetric linear systems. *Numer. Linear Algebra Appl.*, 1:427–447, 1994.

[82] I. E. Kaporin. New convergence results and preconditioning strategies for the conjugate gradient method. *Numerical Linear Algebra with Applications*, 1(2):179–210, 1994.

[83] G. Karypis. *METIS A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices*, 2013.

[84] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48(1):96–129, 1998.

[85] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1999.

[86] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. Society for Industrial and Applied Mathematics, Philadelphia, 1995.

[87] D. E. Keyes and W. D. Gropp. A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation. *SIAM Journal on Scientific and Statistical Computing*, 8(2):s166–s202, 1987.

[88] S. A. Kharchenko and A. Yu. Yeremin. Eigenvalue translation based preconditioners for the gmres(k) method. *Numerical Linear Algebra with Applications*, 2(1):51–77.

[89] L. Yu. Kolotilina. Bounds for eigenvalues of symmetric block jacobi scaled matrices. *Journal of Mathematical Sciences*, 79(3):1043–1047, Apr 1996.

[90] A. N. Krylov. On the numerical solution of the equation by which in technical questions frequencies of small oscillations of material systems are determined. *Izvestiya Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk*, 7:4:491–539, 1931.

[91] S. Lacroix, Y. Vassilevski, and M. F. Wheeler. Decoupling preconditioners in the implicit parallel accurate reservoir simulator (ipars). *Numer. Linear Algebra with Applications*, 8:537–549, 2001.

[92] P. Ladevèze and D. Leguillon. Error estimate procedure in the finite element method and applications. *SIAM J. Numer. Anal.*, 20(3):485–509, 1983.

[93] C. Lanczos. Solution of systems of linear equations by minimized iterations. *J. Res. Nat. Bur. Standards*, 49(1):33–53, 1952.

[94] F. Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 296–303, New York, NY, USA, 2014. ACM.

[95] R. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users' Guide*. Society for Industrial and Applied Mathematics, 1998.

[96] R. Li, Y. Xi, and Y. Saad. Schur complement-based domain decomposition preconditioners with low-rank corrections. *Numerical Linear Algebra with Applications*, 23(4):706–729, 2016.

[97] Z. Li, Y. Saad, and M. Sosonkina. parms: a parallel version of the algebraic recursive multilevel solver. *Numerical Linear Algebra with Applications*, 10(5-6):485–509, 2003.

[98] Y. Liang. *The Use of Parallel Polynomial Preconditioners in the Solution of Systems of Linear Equations*. PhD thesis, Faculty of Informatics of the University of Ulster, 2005.

[99] J. Liesen and P. Tichý. Convergence analysis of krylov subspace methods. *GAMM-Mitteilungen*, 27(2):153–173, 2004.

[100] J. Liu and A. L. Marsden. A robust and efficient iterative method for hyper-elastodynamics with nested block preconditioning. *Journal of Computational Physics*, 383:72 – 93, 2019.

[101] R. Luce and B. I. Wohlmuth. A local a posteriori error estimator based on equilibrated fluxes. *SIAM J. Numer. Anal.*, 42(4):1394–1414, 2004.

[102] G. Mallik, M. Vohralík, and S. Yousef. Goal-oriented a posteriori error estimation for conforming and nonconforming approximations with inexact solvers. working paper or preprint, December 2018.

[103] J. Mandel. On block diagonal and schur complement preconditioning. *Numerische Mathematik*, 58(1):79–93, 1990.

[104] P.J. Matuszyk and K. Boryczko. A Parallel Preconditioning for the Nonlinear Stokes Problem. *Parallel Processing and Applied Mathematics PPAM 2005*, 2006.

[105] D. Meidner, R. Rannacher, and J. Vihharev. Goal-oriented error control of the iterative solution of finite element equations. *J. Numer. Math.*, 17(2):143–172, 2009.

[106] A. Miraçi, J. Papež, and M. Vohralík. A multilevel algebraic error estimator and the corresponding iterative solver with p-robust behavior. working paper or preprint, March 2019.

[107] S. Moufawad. *Enlarged Krylov Subspace Methods and Preconditioners for Avoiding Communication*. PhD thesis, 2014. Thèse de doctorat Mathématiques appliquées UPMC Paris VI.

[108] P. Neittaanmäki and S. Repin. *Reliable methods for computer simulation*, volume 33 of *Studies in Mathematics and its Applications*. Elsevier Science B.V., Amsterdam, 2004. Error control and a posteriori estimates.

[109] Y. Notay. Optimal v-cycle algebraic multilevel preconditioning. *Numerical Linear Algebra with Applications*, 5(5):441–459.

[110] A. S. Odeh. Comparison of solutions to a three-dimensional black-oil reservoir simulation problem (includes associated paper 9741). *Journal of Petroleum Technology*, 33(01):13–25, 1981.

[111] P. Oswald. *Multilevel Finite Element Approximation: Theory & Applications*. Teubner Skripten zur Numerik. Teubner, Stuttgart, 1994.

[112] J. Papež, J. Liesen, and Z. Strakoš. Distribution of the discretization and algebraic error in numerical solution of partial differential equations. *Linear Algebra and its Applications*, 449:89 – 114, 2014.

[113] J. Papež, U. Rüde, M. Vohralík, and B. Wohlmuth. Sharp algebraic and total a posteriori error bounds for h and p finite elements via a multilevel approach. HAL-preprint, December 2017.

[114] J. Papež, Z. Strakoš, and M. Vohralík. Estimating and localizing the algebraic and total numerical errors using flux reconstructions. *Numerische Mathematik*, 138(3):681–721, Mar 2018.

[115] W. Prager and J. L. Synge. Approximations in elasticity based on the concept of function space. *Quart. Appl. Math.*, 5:241–269, 1947.

[116] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Clarendon Press, 1999.

[117] S. Repin. *A posteriori estimates for partial differential equations*, volume 4 of *Radon Series on Computational and Applied Mathematics*. Walter de Gruyter GmbH & Co. KG, Berlin, 2008.

[118] U. Rüde. Fully adaptive multigrid methods. *SIAM J. Numer. Anal.*, 30(1):230–248, 1993.

[119] U. Rüde. *Mathematical and computational techniques for multilevel adaptive methods*, volume 13 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1993.

[120] U. Rüde. Error estimates based on stable splittings. In *Domain decomposition methods in scientific and engineering computing (University Park, PA, 1993)*, volume 180 of *Contemp. Math.*, pages 111–118. Amer. Math. Soc., Providence, RI, 1994.

[121] J. W. Ruge and K. Stüben. *4. Algebraic Multigrid*, pages 73–130.

[122] Y. Saad. Numerical solution of large nonsymmetric eigenvalue problems. *Computer Physics Communications*, 53(1):71 – 90, 1989.

[123] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, second edition, 2000.

[124] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Society for Industrial and Applied Mathematics, 2011.

[125] Y. Saad and M. H. Schultz. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM J. Sci. Statist. Comput.*, 7(3):856–869, 1986.

[126] Y. Saad and M. Sosonkina. Distributed schur complement techniques for general sparse linear systems. *SIAM Journal on Scientific Computing*, 21(4):1337–1356, 1999.

[127] R. Scheichl, R. Masson, and J. Wendebourg. Decoupling and block preconditioning for sedimentary basin simulations. *Computational Geosciences*, 7:295–318, 2003.

[128] J. Scott and M. Tůma. The importance of structure in incomplete factorization preconditioners. *BIT Numerical Mathematics*, 51(2):385–404, Jun 2011.

[129] Y. Shapira. Model case analysis of an algebraic multilevel method. *Numerical Linear Algebra with Applications*, 6(8):655–685.

[130] P. Sonneveld. Cgs, a fast lanczos-type solver for nonsymmetric linear systems. *SIAM journal on scientific and statistical computing*, 10(1):36–52, 1989.

[131] D. C. Sorensen. Implicitly restarted arnoldi/lanczos methods for large scale eigenvalue calculations. In *Parallel Numerical Algorithms*, pages 119–165. Springer, 1997.

[132] G. W. Stewart. A krylov–schur algorithm for large eigenproblems. *SIAM J. Matrix Anal. Appl.*, 23(3):601–614, March 2001.

[133] V. Strassen. Gaussian elimination is not optimal. *Numer. Math.*, 13(4):354–356, August 1969.

[134] K. Stüben. Algebraic multigrid (amg): Experiences and comparisons. *Appl. Math. Comput.*, 13(3-4):419–451, January 1983.

[135] K. Stüben, Clees T., H. Klie, B. Lou, and M. F. Wheeler. Algebraic multigrid methods (AMG) for the efficient solution of fully implicit formulations in reservoir simulation. 2007.

[136] A. Toselli and O. Widlund. *Domains Decomposition Methods: algorithms and theory*. 2005.

[137] L. N. Trefethen and D. Bau. *Numerical linear algebra*, volume 50. Siam, 1997.

[138] A. M. Turing. Rounding-off errors in matrix processes. *The Quarterly Journal of Mechanics and Applied Mathematics*, 1(1):287, 1948.

[139] R. R. Underwood. An approximate factorization procedure based on the block Cholesky decomposition and its use with the conjugate gradient method. Technical report, General Electric Co., Nuclear Energy Div., San Jose, CA., 1976.

[140] H. A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 13(2):631–644, 1992.

[141] R. Verfürth. *A review of a posteriori error estimation and adaptive mesh-refinement techniques*. Teubner-Wiley, Stuttgart, 1996.

[142] P. K. W. Vinsome. ORTHOMIN, an iterative method for solving sparse sets of simultaneous linear equations. In *Proceedings of the Fourth Symposium on Resevoir Simulation*, pages 149–159. Society of Petroleum Engineers of AIME, 1976.

[143] M. Vohralík. Residual flux-based a posteriori error estimates for finite volume and related locally conservative methods. *Numerische Mathematik*, 111(1):121–158, 2008.

[144] M. Vohralík and S. Yousef. A simple a posteriori estimate on general polytopal meshes with applications to complex porous media flows. *Computer Methods in Applied Mechanics and Engineering*, 331:728 – 760, 2018.

[145] J. R. Wallis, R. P. Kendall, and T. E. Little. Constrained residual acceleration of conjugate residual methods. 1985.

[146] A. Y. Yeremin and L. Y. Kolotilina. Factorized sparse approximate inverse preconditionings I: Theory. *SIAM J. Matrix Anal. Appl.*, 14:45–58, 1993.

[147] F. Zhang. *The Schur complement and its applications*, volume 4. Springer Science & Business Media, 2006.

[148] O. C. Zienkiewicz and J. Z. Zhu. A simple error estimator and adaptive procedure for practical engineering analysis. *Internat. J. Numer. Methods Engrg.*, 24(2):337–357, 1987.