N° Ordre : 152

Habilitation à Diriger des Recherches

Spécialité : Informatique

On the Numerical Solution of Partial Differential Equations: Iterative Solvers for Parallel Computers

Sur la Résolution Numérique d'Equations aux Dérivées Partielles: Solveurs Itératifs pour Calculateurs Parallèles

présentée le 13 Septembre 2000 à

l'Institut National Polytechnique de Toulouse

par

Luc GIRAUD
CERFACS

devant le Jury composé de :

T. F. Chan	Professor at UCLA	Rapporteur
Q. V. Dinh	Dassault Aviation	
I. S. Duff	Project Leader at CERFACS	
	Group Leader at Rutherford Appleton Laboratory	
G. H. Golub	Professor at Stanford University	
G. Meurant	CEA	
P. Le Tallec	Professeur à l'Université Paris IX Dauphine	Président
	Professeur à l'Ecole Polytechnique	Rapporteur
P. Spitéri	Professeur à l'ENSEEIHT	
H. A. van der Vorst	Professor at the University of Utrecht	Rapporteur

CERFACS report: TH/PA/00/64

Frédéric Mistral
8 Septembre 1860 (Maillance)
25 Mars 1914 (Maillance)

Bastò, pèr iéu, sus la mar de l'istòri,
Fuguères tu, Provènço, un pur simbèu,
Un miramen de glòri e de vitòri
Que, dins l'oumbrun di siècle transitòri,
Nous laisso vèire un eslùci dóu Béu.

Il suffit, pour moi, sur la mer de l'histoire,
Tu fus, Provence, un pur symbole,
Un mirage de gloire et de victoire
Qui, dans la transition ténébreuse des siècles,
Nous laisse voir un éclair de Beauté.

Extrait de Lis óulivado - 1912

Qu gagno tèms, gagno tout.

Celui qui gagne du temps, gagne tout.

Tèms, fa chanja, madura, óublida e mouri.

Le temps fait changer, mûrir, oublier et mourir.

Proverbes provençaux sur le temps extraits de Lou Tresor dóu Félibrige - 1886

Remerciements - Acknowledgements

Back in 1992, when I started to work on non-overlapping domain decomposition with Ray Tuminaro I had the chance to meet Tony Chan for the first time during his visit at CERFACS. The discussions we had at that time were very stimulating and I am delighted that Tony Chan has accepted to act as a referee for my Habilitation.

Je remercie Monsieur Quang H. Dinh d'avoir participé à mon jury. Ce fut une grande satisfaction que de développer une collaboration industrielle fructueuse avec Dassault Aviation sur un domaine qui a permis d'illustrer la complémentarité des compétences des équipes du CERFACS.

Iain Duff, my sincere gratitude for the freedom and support you gave me to develop my research in your group at CERFACS. Being involved in the management of industrial contracts and management of the Parallel Algorithm Project under your supervision was an immeasurably professionnal experience.

My sincere thanks go to Professor Gene Golub for him taking part in my jury. It was for me an immense honour and a great pleasure.

Je remercie sincèrement Monsieur Gérard Meurant pour l'intérêt qu'il a porté à mon travail. Ses encouragements et sa bienveillance tout au long de la préparation de la thèse de Luiz M. Carvalho, premier doctorant que j'ai encadré scientifiquement, furent très stimulants. Je suis très heureux et honoré qu'il ait accepté de participer au jury.

C'est un grand honneur que d'avoir pour rapporteur et président du jury Monsieur Patrick Le Tallec. La possibilité que j'ai eue de travailler avec Patrick Le Tallec dans le cadre d'une collaboration CERFACS-INRIA a été une chance et une expérience professionnelle et scientifique très enrichissante. Je tiens à le remercier à nouveau très sincérement.

Monsieur Pierre Spitéri, d'abord en tant qu'enseignant à l'ENSEEIHT, puis directeur de thèse et enfin correspondant scientifique à l'Institut National Polytechnique de Toulouse pour cette Habilitation, vous m'avez toujours exprimé votre soutien et vos encouragements bienveillants. Lorsqu'il y a plus de dix ans (déjà !) vous m'avez proposé de préparer une thèse sur le thème du calcul parallèle et de la résolution d'équations aux dérivées partielles, je n'imaginais pas que cela me conduirait un jour à présenter une Habilitation dans le même domaine. Pour tout cela je tiens à vous exprimer toute ma gratitude.

I would like to express my deep thanks to Professor Henk van der Vorst who accepted to act as a referee for my Habilitation; getting his feedback on my research work is a great privilege.

Finally, I would like to add that I have been very proud to defend my Habilitation in front of a such prestigious jury. It was for me an immense honour.

Un grand merci au CERFACS, à tout son personnel administratif et informatique si compétent, disponible et diligent, à tous les membres du Projet Algorithmes Parallèles pour l'ambiance de travail qui y règne. Avoir eu l'opportunité de travailler dans une centre de recherche international et dynamique m'a permis de rencontrer, parfois collaborer et souvent apprécier de nombreuses personnes issues d'horizons divers et variés. Ceci a été pour moi un source extraordinaire d'enrichissement culturel et scientifique. Nombre des personnes que j'y ai rencontré ont joué un rôle important dans ma carrière professionnelle mais il me paraît difficile voire impossible d'en dresser une liste exhaustive; plutôt que de prendre le risque d'un oubli maladroit je préfère me replier derrière un merci collectif mais sincère. Comme toute règle sitôt ennoncée doit posséder une exception, je m'empresse d'en faire une pour Valérie Frayssé, collègue de travail mais amie avant tout, que je dois remercier pour avoir, ces dernières années, partagé avec moi la gestion du quotidien du Projet Algorithmes Parallèles. Enfin je remercie très chaleureusement ma famille et mes amis qui m'ont accompagné.

On the Numerical Solution of Partial Differential Equations: Iterative Solvers for Parallel Computers

Abstract
Numerical simulations of complex physical phenomena often require the use of parallel computers. In order to fully benefit from the capabilities of those computers, new algorithms have to be designed. In addition to the mathematical properties of the problems to be solved, the features of the target computers should be taken into account when designing those algorithms. We focus on algorithms for the numerical solution of linear systems arising from the discretization of PDEs and on their efficient implementations on parallel distributed platforms. In particular we consider domain decomposition techniques with and without overlap for finite element discretization and present some results using sparse approximate inverse approaches for dense linear systems arising from boundary element method in electromagnetism.

Keywords: domain decomposition, two-level preconditioner, sparse approximate inverse preconditioner, Krylov methods, parallel distributed computing.

Sur la Résolution Numérique d'Equations aux Dérivées Partielles: Solveurs Itératifs pour Calculateurs Parallèles

Résumé
Les besoins croissants en puissance de calcul pour la simulation numérique de phénomènes complexes ont conduit à l'utilisation quasi systématique de calculateurs parallèles. Afin d'exploiter efficacement les capacités de ces ordinateurs une nouvelle algorithmique a dû être développée. En plus des caractéristques du problème mathématique à résoudre les nouvelles méthodes numériques doivent prendre en compte les particularités architecturales des calculateurs sur lesquels elles seront inplantées. Ce document est consacré à la présentation de méthodes itératives pour la résolution de systèmes linéaires issus de la discrétisation de problèmes aux dérivées partielles ainsi qu'à la description de leur implantation sur machines parallèles à mémoire distribuée. Nous considérons en particulier des méthodes de décomposition de domaines avec ou sans recouvrement pour des discrétisations de type éléments finis et présentons quelques résultats obtenus en utilisant des techniques d'inverse approchée pour la résolution de systèmes linéaires denses issus de discrétisation par éléments frontières en électromagnétisme.

Mots-clés : méthodes de décomposition de domaine, préconditionneurs à deux niveaux, préconditionneurs creux par inverse approchée, méthodes de Krylov, calcul parallèle distribuè.

Contents

Part I

Introduction

"With the development of new kinds of equipment of greater capacity, and particularly of greater speed, it is almost certain that new methods will have to be developed in order to make fullest use of the new equipment. It is necessary not only to design machines for the mathematics, but also to develop a new mathematics for the machines" declared D. Hartree in 1952 when ENIAC (Electronic Numerical Integrator and Computer) was just built. This vision has been the scope of an extensive research work in the last decades and further extended to include parallel computers. The study and development of algorithms for high performance computers has been my research topic first as a PhD student in the ENSSEIHT-IRIT Lab, then as Post-Doc and Senior researcher at CERFACS in the Parallel Algorithms Project.

CERFACS is a quite unique place where basic research and applied research through industrial collaborations co-exist and interact, enabling cross fertilization between academic and industrial communities. The CERFACS researchers time is split between those two activities enabling to work on several topics that, although connected, are sometimes only loosely coupled. Let me illustrate this feature through my own experience where the links among all the activities are high performance computing or linear algebra. Within these collaborations, my work addresses various topics ranging from computer science concerns such as

- the impact evaluation of the new computer architecture on scientific computing codes from Aerospatiale [51] or CNES [3, 82],
- the development of a parallel distributed fast Poisson solver and the definition of a parallelization strategy for a meso-scale weather forecasting vectorial code for Météo-France [81, 108],
- the porting on distributed memory platforms of the Computational Fluid Dynamic (CFD) code N3S in the framework of an EC HPCN-ESPRIT project [54, 83],
- or the development of an object oriented software to manage a pool of parallel tasks on heterogeneous networks of computers addressing the load balancing and fault tolerance capabilities in few european ESPRIT projects [8, 9, 109];

to more numerical analysis issues including

- the study of efficient parallel preconditioners for implicit schemes in CFD in a joint work with Dassault-Aviation [40],
- the study of various Krylov solvers for complex symmetric non Hermitian matrices with Thomson-LCR [50],
- the development of the Block-QMR [74] variant for J-symmetric matrices to handle multiple right hand sides in electromagnetic applications with Aerospatiale [67],
- the study of parallelizable preconditioners for dense linear systems in a joint work with Aerospatiale [16].

To illustrate how cross fertilization sometimes goes beyond the simple collaboration between the two communities, let me describe the following example. In a joint work with CNES [68], we have developed a GMRES implementation for complex matrices. This code was then further improved to deal with any type of arithmetics and to satisfy some software quality requirements enabling its efficient use on sequential and shared or distributed memory parallel computers. The resulting packages [69, 70] have been put in the public domain, `www.cerfacs.fr/algor/Softs/`, with a non-commercial license agreement. It is regularly downloaded by many researchers working in various areas ranging from geophysics or ocean modeling to theoretical physics. In particular, the complex version has been recently integrated in a public domain circuit simulator developed by [] at

Research Bell Lab. In addition to the enjoyable research aspects related to these various projects, learning about their management is also a valuable component as any successful collaboration not only requires intensive scientific work but also some administrative management.

In this manuscript, we will partially illustrate those two aspects of the research at CERFACS but we only report studies conducted on some parallel and parallelizable numerical iterative techniques for the solution of linear systems arising from the discretization of partial differential equations (PDE). These linear systems are sparse for finite difference or finite element discretizations. Their parallel solution is often tackled via domain decomposition techniques that are well adapted for distributed memory computers. When boundary elements discretization are preferred, because they are better suited to represent the physical problems as in some electromagnetic or acoustic applications, the resulting linear system is dense. In this latter situation preconditioned Krylov solvers are a promising alternative to the classical method of choice, that is the Gaussian elimination, provided we have fast matrix-vector multiplications and robust and effcient preconditioners.

When dealing with linear systems arising from PDE the scalability of the numerical method and the scalability of its parallel implementation are key aspects to address when targeting the solution of large problems. In this respect, we consider those two aspects in separate parts of this document. Part II describes the numerical behavior of the numerical techniques we are interested in and the performance of their parallel implementation is discussed in Part III.

This document is organized as follows. In Part II, we describe in Chapter 1 the numerical behavior of asynchronous iterations that might constitute an original alternative to load balancing in order to minimize the idle time of processors for optimizing the throughput of parallel computing resources. Chapter 2 is devoted to two-level non-overlapping domain decomposition for elliptic self adjoint operators. We describe parallel preconditioners that can be written as the sum of a symmetric positive definite matrix, aiming at capturing the local behavior of the operator plus a low rank update involving a coarse problem, that intends to represent the global behavior of the elliptic equation. To illustrate the industrial and inter-disciplinary collaborations at CERFACS we present in Chapter 3 some investigations using overlapping domain decomposition methods in an industrial code for CFD. Finally, Chapter 4 presents some sparse approximate inverse preconditioners for the solution of dense matrices arising in computational electromagnetics. This work is motivated by the observation that since the inverse of the inverse of a sparse matrix is sparse, then there are classes of dense matrices for which a sparse approximate inverse might be an appropriate preconditioner. Part III is devoted to the parallel implementation and performance of all the numerical methods studied in Part II, except the parallelization of the approximate inverse preconditioner which we plan to address in a near future. In particular a study of the numerical scalability of the approximate inverse preconditioner in the context of electromagnetism applications, that is mandatory before addressing the parallel implementation issues, is presented in Part IV which is devoted to conclusions and future works.

All the work presented in this document was, and in some case is still, developed in collaboration with other researchers. The asynchronous iterations work [80, 85] was performed with P. Spitéri (ENSEEIHT-IRIT) during my PhD; the domain decomposition in part with R. S. Tuminaro (Sandia National Lab.) [89], with L. M. Carvalho (University of Rio Janeiro) and P. Le Tallec (Université Paris IX Dauphine) [39], with L. M. Carvalho and G. Meurant (CEA) [38]. Part of this work was carried out in the framework of L. M. Carvalho's PhD thesis [35] at CERFACS. The investigations on overlapping domain decomposition were conducted in collaboration with G. Chevalier (CERFACS), F. Chalot and Q. V. Dinh (Dassault Aviation) [40]. Lastly the approximate inverse preconditioners for dense matrices were first studied in the framework of an industrial collaboration in a joint work with G. Alléon (CCR Aerospatiale) and M. Benzi (formerly at CERFACS and now Los Alamos National Lab.) [2, 16] and further developed with B. Carpentieri (CERFACS)

and I. S. Duff (CERFACS - RAL) [33, 34]. B. Carpentieri is currently completing his PhD on this latter subject.

Part II

Some Numerical Solvers for Partial Differential Equations

Chapter 1

Asynchronous methods

1.1 Introduction

With the advent of parallel computers, many new algorithms were designed or rediscovered to fully exploit the new architectures. An dominating concept in the design of parallel algorithms is that the work should be equally spread among the processors in order to reduce the idle time resulting from unbalance granularity of the concurrent tasks. This constraint is widely integrated as a requirement to design efficient parallel algorithms and governs for instance the graph partitioning algorithms [102, 110] used for parallel sparse matrix computation or domain decomposition. In contrast to load balancing, the idea of asynchronous methods is to avoid processor idle time by eliminating as much as possible synchronization points. The price to pay for this freedom is that some processors will perform extra computations that are expected to be beneficial when the load is unbalanced or when the communication between the processors is slow.

Since the pioneer paper on this method [46] many authors have intensively studied the theory and the applications of asynchronous iterations. The convergence analysis of the asynchronous iterations can be done using different techniques; we mention for instance the analysis based on some contraction properties in appropriate vectorial norm [46, 128, 139] or the one using order intervals [65, 129].

Let us also mention some papers where the application of asynchronous iterations to different areas is discussed: the solution of partial differential equations [5]; to overlapping domain decomposition [77, 148]; to inverse problems in geophysics and oil exploration [133]; to electrical power network [10]; to network flow [151]; to convex programming [149], and other optimization [64] and nonlinear problems [150] and to singular systems of linear equations [134]. In addition notice that because of the asynchronous behavior of the algorithms special attention should be paid to the implementation of the stopping criteria [63, 144]. Finally for surveys of asynchronous iterative methods we refer to [19, 20] and [78] where part of the above list was found.

This chapter is organized as follows. In Section 1.2 we first recall the mathematical model commonly used to describe the asynchronous iterations and recall a general convergence theorem based on vectorial norm contraction properties. We also give a characterization of the contraction matrix using accretive properties of the submatrices that naturally appears when block asynchronous algorithms are considered [85].

1.2 Mathematical model and theoretical framework

1.2.1 Mathematical models

Consider the following fixed point equation $x^* = F(x^*)$ defined on E a product of Banach spaces, i.e. $E = E_1 \times ... \times E_m$.

To describe asynchronous iterations we follow [19, 46, 128] and first define the policy $s(p)$ and a series of delays $d(p)$.

Definition II.1.1 *A policy is defined by a series $s(p)$ such that:*

$$\forall p \in I\!N, \quad s(p) \subset \{1, ..., m\}, \; s(p) \neq \emptyset, \tag{II.1.1}$$

$$\forall k \in \{1, ..., m\} \text{ the set } \; \{p \in I\!N \text{ s.t. } k \in s(p)\} \text{ is infinite.} \tag{II.1.2}$$

Definition II.1.2 *A series of delays $d(p)$ is defined by*
$\forall p \in I\!N, r(p) = (r_i(p))_{i=1,m} \in I\!N^m$
$\forall p \in I\!N, \forall k \in \{1, ..., m\}$ *the application: $p \to c_k(p) = p - r_k(p)$ is a non-decreasing function of p and satisfies:*

$$c_k(p) \geq 0 \text{ and } \; c_k(p) = p \;\; \forall k \in s(p), \tag{II.1.3}$$

$$\lim c_k(p) = \infty. \tag{II.1.4}$$

Using the two above definitions the asynchronous iterations can be defined as follows.

Definition II.1.3 *Let $x^0 \in E$ we consider the series of iterates defined by:*

$$\forall p \in I\!N, \; \forall k \in \{1, ..., m\}, \quad x_k^{p+1} = \begin{cases} x_k^p & \text{if} \quad k \notin s(p) \\ F_k(w) & \text{if} \quad k \in s(p) \end{cases} \tag{II.1.5}$$

where $w = (x_\ell^{c_\ell(p)})_{\ell=1,m}$, $w \in E$.

The policy series $s(p)$ enables to define the components of the iterate that will be updated at iteration p, the series $d(p)$ enables to model the asynchronism and $c_\ell(p)$ indicates the iteration when the ℓ^{th} component just read to perform the p^{th} iteration was computed. The hypothesis (II.1.3) indicates that all the components in $s(p)$ will be updated at the p^{th} iteration, (II.1.4) indicates that the computation proceeds and (II.1.2) that no components fails to be updated as time goes on. Finally the non decreasing property of the application $c_k(.)$ tells that one always consider the latest computed entries available to perform the update given by (II.1.5).

In the sequel we will only consider the situation where $F_k(w)$ defines a relaxation update like Gauss-Seidel or SOR (Successive Over Relaxation). Note that Definition II.1.3 includes as special case the classical sequential or synchronous stationary methods. For linear relaxation schemes, $\forall p \;\; s(p) = \{1, ..., m\}$ models the block-Jacobi algorithm, while $\forall p \;\; s(p) = \{ \mod (p, m) + 1\}$ corresponds to the block Gauss-Seidel method.

1.2.2 A convergence analysis framework

A general convergence theorem for the asynchronous iterations is the following [128, 138].

Theorem II.1.1 *Let F be an application from $D(F) \subset E$, $D(F) \neq \emptyset$ into $D(F)$.*
Let assume that

F has a fixed point $x^* \in D(F)$,

F is a J-contraction with respect to the fixed point x^*, that is there exists a nonnegative matrix $J \in \mathbb{R}^{m\times m}$ with $\rho(J) < 1$ such that $\forall x \in D(F)$

$$\begin{pmatrix} ||F(x_1) - F(x_1^*)||_1 \\ \vdots \\ ||F(x_m) - F(x_m^*)||_m \end{pmatrix} < J. \begin{pmatrix} ||x_1 - x_1^*||_1 \\ \vdots \\ ||x_m - x_m^*||_m \end{pmatrix},$$

where the inequality in $\mathbb{R}^m$ is componentwise, $\rho(J)$ is the spectral radius of the matrix J and $||.||_i$ is the norm defined on the Banach space E_i.

Then (II.1.5) *defines x^p for all $p \in \mathbb{N}$ such that $x^p \in D(F)$ and the iterates x^p converge to x^* the fixed point of F.*

To establish a convergence theorem for asynchronous iterations we consider in the sequel nonlinear algebraic problems of the form:

$$Ax + \Lambda(x) - b = 0 \tag{II.1.6}$$

where A is a $n \times n$ matrix, x and b are vectors and Λ is a diagonal operator possibly multivoque. We now consider a decomposition of (II.1.6) into m blocks, that is

$$\forall k \in \{1, ..., m\} \;\; A_{kk}x_k + \Lambda_k(x_k) - b_k + \sum_{j\neq k} A_{kj}x_j = 0.$$

Theorem II.1.2 *Let*

- *$\forall k \in \{1, ..., m\}$ the matrix A_{kk} be strongly accretive with constant m_{kk}, that is for all $x_k \in E_k \equiv \mathbb{R}^{n_k}$ there exists a dual $l_k(x_k)$ of x_k such that*

$$\langle A_{kk}x_k, l_k(x_k)\rangle \geq m_{kk}||x_k||_k^2.$$

Here $\langle.\rangle$ denotes the bilinear form between $(\mathbb{R}^{n_k}, ||.||_k)$ as a Banach space and its dual $(\mathbb{R}^{n_k}, ||.||_k)^*$ where $l_k(x_k)$ is an element of $(\mathbb{R}^{n_k}, ||.||_k)^*$ with

$$||l_k(x_k)||_k^* = ||x_k||_k \;\text{ and }\; \langle l_k(x_k), x_k\rangle = ||x_k||_k^2.$$

- *$\forall (j,k) \in \{1, ..., m\}^2$, $j \neq k$, m_{jk} be the norm of the matrix A_{jk},*
- *$\forall (j,k) \in \{1, ..., m\}^2$, Λ_k be an increasing application,*
- *the matrix J be the matrix with zero diagonal entries and off-diagonal entries equal to $\frac{m_{kj}}{m_{kk}}$ and suppose that J is a contraction matrix.*

Then the parallel synchronous and asynchronous block algorithms associated with the $m \times m$ decomposition defined by (II.1.5) *converge to x^* solution of* (II.1.6).

Proof See [85].

Remark II.1.1 *In practice for J it is enough to show that $\rho(J) < 1$ since it is clearly a nonnegative matrix. The contraction property is true in particular if the related matrix $\bar{M}$ whose diagonal entries are m_{kk} and off-diagonal entries $-m_{jk}$ is a M-matrix (see [131]).*

Remark II.1.2 *Notice that if A in* (II.1.6) *is a M-matrix and Λ is an increasing diagonal operator, then the choice of the point wise decomposition, i.e. $m = n$, enables to show that the point wise asynchronous algorithm converges and consequently any block asynchronous algorithm also converges.*

1.3 Numerical experiments

To illustrate the numerical behavior of asynchronous iterations we consider the discretized and linearized Hamilton-Jacobi-Bellman problem and refer to [85] where results on other nonlinear problems like the obstacle problem and a nonlinear diffusion problem are reported. We recall that the Hamilton-Jacobi-Bellman equations appears in different area like economics, stochastic control, ...

In this section we consider the following discretized Hamilton-Jacobi-Bellman equation:

$$\text{Find x such that } \max_{i=1,2}(A_i x - b_i) = 0. \tag{II.1.7}$$

where the matrices A_i are the discretization of the following elliptic operators defined on the unit square with homogeneous Dirichlet boundary conditions:

$$\begin{cases} \mathcal{A}_1 & = -(\dfrac{\partial^2}{\partial x^2} + \dfrac{\partial^2}{\partial y^2} + \dfrac{1}{2}\dfrac{\partial^2}{\partial x \partial y}), \\ \mathcal{A}_2 & = -(\dfrac{1}{2}\dfrac{\partial^2}{\partial x^2} + \dfrac{\partial^2}{\partial y^2} + \dfrac{1}{10}\dfrac{\partial^2}{\partial x \partial y}). \end{cases}$$

The discretized problem (II.1.7) is solved using the Howard iterative scheme [104] that consists in computing the new iterate x^{p+1} such that it is the solution of the linear system

$$A(x^p)x^{p+1} = b(x^p), \tag{II.1.8}$$

where the i^{th} row of $A(x^p)$ and the i^{th} entry of the vector $b(x^p)$ are defined by the i^{th} row of the matrix A_k, respectively the i^{th} entry of the vector b_k if k is such that $(A_k x^p - b_k)_i > (A_j x^p - b_j)_i$.

If the operator $\mathcal{A}_i$ are discretized using finite differences on a uniform grid, the discretization matrices A_i are M-matrices then $\forall x^p \;\; A(x^p)$ is also a M-matrix and Remark II.1.2 enables to establish the convergence of asynchronous iterations on the linear system (II.1.8) for any decomposition of the unit square into subdomains.

The purpose of the reported experiments is to illustrate that introducing some asynchronism in the relaxation schemes does not deteriorate, and in some cases even improve, the numerical behavior of the sequential code in terms of number of iterations. Asynchronous iterations were implemented on shared and distributed memory computers. For all the experimental results reported in this section, the convergence of the relaxation schemes is attained when the 2-norm of the residual is less than 10^{-6}, the initial guess x_0 for the relaxation iterations was the null vector. In Table II.1.1 we report the number of iterations observed on a network of Transputer (a loosely coupled parallel platform), and depict in Table II.1.2 the numerical behavior observed on an Alliant FX/80, a vectorial shared memory computer. The linear solver for (II.1.8) is a block SOR method, where each block is associated with one line of the grid. The asynchronism is obtained by decomposing the linear system (II.1.8) into m blocks of rows and allocating one of these m blocks to each processor. Notice that in terms of mesh decomposition this corresponds to a partition of the grid points into strips, i.e. 1 D decomposition. Because we consider asynchronous algorithms for each run we give both the minimum and the maximum number of iterations performed by the processors.

Two parameters play an important role in the numerical behavior of the asynchronous relaxation schemes that are the subdomain decomposition, that might give rise to unbalance, and the over-relaxation parameter of the SOR scheme. There are no theoretical study nor heuristic that enable to define their optimal values. In this respect all the number of iterations reported in the above tables are the optimal we observed for a given number of processors varying both the decomposition and the relaxation parameter. Although no numerical experiments are reported to illustrate this fact, we mention that the convergence rate of the asynchronous SOR relaxations is

Number of processors	Number of iterations (min/Max)
1	241
2	244/247
4	241/247
8	145/239

Table II.1.1: Number of iterations of the asynchronous iterations on a network of Transputer.

Number of processors	Number of iterations (min/Max)
1	241
2	255/258
4	238/271
8	144/228

Table II.1.2: Number of iterations of the asynchronous iterations on an Alliant FX/80.

often very sensitive to those two parameters. In addition we mention that the best combination of those parameters is also dependent on the target computer as the communication speed depends on the computer influences the numerical behavior. Nevertheless it is amusing to notice that on 8 processors for both computers the sequence of iterates generates by the asynchronous iterations enables to converge in less iterations than the sequential code.

Chapter 2

Non-overlapping domain decomposition methods

2.1 Introduction

In the recent years, there has been an important development of domain decomposition algorithms for solving numerically elliptic partial differential equations. Elliptic problems are challenging since their Green's functions are global: the solution at each point depends upon the data at all other points. Nowadays some methods possess optimal convergence rates for given classes of elliptic problems. It can be shown that the condition number of the associated preconditioned systems is independent of the number of subdomains and either independent of or logarithmically dependent on the size of the subdomains. That optimality and this quasi-optimality properties are often achieved thanks to the solution of a coarse problem defined on the whole physical domain. Through the use of coarse spaces, this approach captures the global behavior of the elliptic equations. Various domain decomposition techniques, from the eighties and nineties, have suggested different global coupling mechanisms and various combinations between them and the local preconditioners. In the framework of non-overlapping domain decomposition techniques, we refer for instance to BPS (Bramble, Pasciak and Schatz) [26], Vertex Space [60, 146], and to some extend Balancing Neumann-Neumann [116, 118, 120], as well as FETI [66, 122], for the presentation of major two-level preconditioners. We refer to [44, 147] and [136] for a more exhaustive overview of domain decomposition techniques.

This global coupling is critical for the numerical scalability of the preconditioners. In particular, it has been shown in [26] that, when applying the original BPS technique to a uniformly elliptic operator, the preconditioned system has a condition number

$$\kappa(M_{BPS}S) = \mathcal{O}(1 + \log^2(H/h)), \tag{II.2.1}$$

where h is the mesh size, H largest diameter of the subdomains and $\kappa(A)$ is the condition number of the matrix A. This implies that the condition number depends only weakly on the mesh spacing and on the number of subdomains. Therefore, such a preconditioner is numerically scalable and appropriate for large systems of equations solved on large processor systems.

Similarly to BPS, we consider a class of preconditioners described in a generic way as:

$$M = M_{local} + M_{global}. \tag{II.2.2}$$

In Section 2.2 we describe several alternatives to define M_{local} and in Section 2.3 we propose a set of coarse components to be used for M_{global}.

In this chapter, we consider the following 2^{nd} order self-adjoint elliptic problem on an open polygonal domain Ω included in $\mathbb{R}^2$:

$$\left\{ \begin{array}{rcll} -\frac{\partial}{\partial x}(a(x,y)\frac{\partial v}{\partial x}) - \frac{\partial}{\partial y}(b(x,y)\frac{\partial v}{\partial y}) & = & F(x,y) & \text{in} \quad \Omega, \\ v & = & 0 & \text{on} \quad \partial\Omega, \end{array} \right. \tag{II.2.3}$$

where $a(x,y)$, $b(x,y) \in \mathbb{R}^2$ are bounded positive functions on Ω. We assume that the domain Ω is partitioned into N non-overlapping subdomains $\Omega_1, \ldots, \Omega_N$ with boundaries $\partial\Omega_1, \ldots, \partial\Omega_N$; this defines a coarse mesh, τ^H, with mesh size H being the largest diameter of the subdomains. We discretize (II.2.3) either by finite differences or finite elements resulting in a symmetric and positive definite linear system with sparse possibly unstructured matrix

$$Au = f.$$

Let B be the set of all the indices of the discretized points which belong to the interfaces between the subdomains. Grouping the points corresponding to B in the vector u_B and the ones corresponding to the interior I of the subdomains in u_I, we get the reordered problem:

$$\begin{pmatrix} A_{II} & A_{IB} \\ A_{IB}^T & A_{BB} \end{pmatrix} \begin{pmatrix} u_I \\ u_B \end{pmatrix} = \begin{pmatrix} f_I \\ f_B \end{pmatrix} . \tag{II.2.4}$$

Eliminating u_I from the second block row of (II.2.4) leads to the following reduced equation for u_B:

$$Su_B = f_B - A_{IB}^T A_{II}^{-1} f_I , \qquad \text{where} \qquad S = A_{BB} - A_{IB}^T A_{II}^{-1} A_{IB} \tag{II.2.5}$$

is the Schur complement of the matrix A_{II} in A, and is usually referred to as the Schur complement matrix.

The matrix S inherits from A the symmetric positive definiteness property. Therefore we use preconditioned conjugate gradient iterations for solving (II.2.5).

2.2 Local preconditioners

To describe the preconditioners, we need to define a partition of B, the set of interface points. Let $\{v_l\}$ be the singleton sets that contain one index related to one cross point and let $V = \bigcup_j \{v_j\}$ be the set with all those indices.

If $i \neq j$, $(i,j) \in \{1,2,\ldots,N\}^2$ and i and j are such that Ω_i and Ω_j are neighboring subdomains (i.e. $\partial\Omega_i$ and $\partial\Omega_j$ share at least one edge of the mesh), then we can define each edge E_k by

$$E_k = (\partial\Omega_i \cap \partial\Omega_j) \backslash V.$$

We can thus describe the set B as

$$B = (\bigcup_k E_k) \cup V, \tag{II.2.6}$$

that is a partition of the interface B into edges E_ℓ and the set of vertices V.

In Figure II.2.1, we depict an internal subdomain Ω_i with its edge interfaces E_m, E_g, E_k, E_ℓ and vertex points as v_l that define $\Gamma_i = \partial\Omega_i \backslash \partial\Omega$. Let $R_{\Gamma_i} : \Gamma \to \Gamma_i$ be the canonical pointwise restriction which maps full vectors defined on Γ into vectors defined on Γ_i, and let $R_{\Gamma_i}^T : \Gamma_i \to \Gamma$

be its transpose. For a stiffness matrix A arising from a finite element discretization, the Schur complement matrix (II.2.5) can also be written as:

$$S = \sum_{i=1}^{N} R_{\Gamma_i}^T S^{(i)} R_{\Gamma_i},$$

where

$$S^{(i)} = A_{\Gamma_i}^{(i)} - A_{i\Gamma_i}^T A_{ii}^{-1} A_{i\Gamma_i} \tag{II.2.7}$$

is referred to as the local Schur complement associated with the subdomain Ω_i. $S^{(i)}$ involves submatrices from the local stiffness matrix $A^{(i)}$, defined by

$$A^{(i)} = \begin{pmatrix} A_{ii} & A_{i\Gamma_i} \\ A_{i\Gamma_i}^T & A_{\Gamma_i}^{(i)} \end{pmatrix}. \tag{II.2.8}$$

The matrix $A^{(i)}$ corresponds to the discretization of Equation (II.2.3) on the subdomain Ω_i with

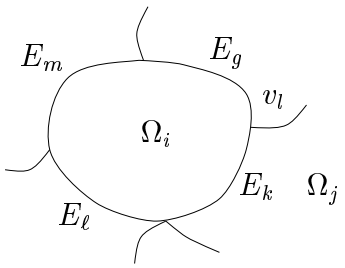


FIGURE II.2.1: An internal subdomain.

Neumann boundary condition on Γ_i and A_{ii} corresponds to the discretization of Equation (II.2.3) on the subdomain Ω_i with homogeneous Dirichlet boundary conditions on Γ_i. In a parallel distributed memory environment, where each subdomain is assigned to one processor, all the local Schur complement matrices can be computed simultaneously by all the processors and the complete Schur matrix S defined by (II.2.5) is never fully assembled.
The local Schur complement matrix, associated with the subdomain Ω_i depicted in Figure II.2.1, is dense and has the following block structure:

$$S^{(i)} = \begin{pmatrix} S_{mm}^{(i)} & S_{mg} & S_{mk} & S_{m\ell} & S_{mV}^{(i)} \\ S_{gm} & S_{gg}^{(i)} & S_{gk} & S_{g\ell} & S_{gV}^{(i)} \\ S_{km} & S_{kg} & S_{kk}^{(i)} & S_{k\ell} & S_{kV}^{(i)} \\ S_{\ell m} & S_{\ell g} & S_{\ell k} & S_{\ell\ell}^{(i)} & S_{\ell V}^{(i)} \\ S_{Vm}^{(i)} & S_{Vg}^{(i)} & S_{Vk}^{(i)} & S_{V\ell}^{(i)} & S_{VV}^{(i)} \end{pmatrix},$$

where V is the set of vertices v_l of Ω_i. The first four diagonal blocks represent the local coupling between nodes on an edge interface introduced by the subdomain Ω_i and are only contributions to the diagonal blocks of the complete Schur complement matrix S. For instance, the diagonal block of the complete matrix S associated with the edge interface E_k in Figure II.2.1 is $S_{kk} = S_{kk}^{(i)} + S_{kk}^{(j)}$.

Assembling each diagonal block of the local Schur complement matrices and the blocks associated with the vertices, we obtain the local assembled Schur complement, that is:

$$\tilde{S}^{(i)} = \begin{pmatrix} S_{mm} & S_{mg} & S_{mk} & S_{m\ell} & S_{mV} \\ S_{gm} & S_{gg} & S_{gk} & S_{g\ell} & S_{gV} \\ S_{km} & S_{kg} & S_{kk} & S_{k\ell} & S_{kV} \\ S_{\ell m} & S_{\ell g} & S_{\ell k} & S_{\ell\ell} & S_{\ell V} \\ S_{Vm} & S_{Vg} & S_{Vk} & S_{V\ell} & S_{VV} \end{pmatrix},$$

which corresponds to the restriction of S to the unknowns associated with the interface Γ_i of Ω_i.

The new local preconditioners can be described using a set of canonical restriction operators. Let U be the algebraic space of nodal vectors where the Schur complement matrix is defined and $(U_i)_{i=1,p}$ a set of subspaces of U such that:

$$U = U_1 + ... + U_p.$$

Let R_i be the canonical pointwise restriction of nodal values defined on U_i. Its transpose extends grid functions in U_i by zero to the rest of U. Using the above notation, we can define a wide class of block preconditioners by

$$M_{loc} = \sum_{i=1}^{p} R_i^T M_i^{-1} R_i, \tag{II.2.9}$$

where

$$M_i = R_i S R_i^T. \tag{II.2.10}$$

The properties of the operators (II.2.9) and (II.2.10) are given by the following lemma:

Lemma II.2.1 *If the operator R_i^T is of full rank and if S is symmetric and positive definite, then the matrix M_i, defined in Equation (II.2.10), and the matrix M_{loc} defined in Equation* (II.2.9) *are symmetric and positive definite.*

Proof
The proof can be done in two steps. We first show that M_i^{-1} is symmetric positive definite (SPD) then that M_{loc} is SPD.
Let $< ., . >$ denotes the scalar product associated with the 2-norm.

- M_i^{-1} is SPD is equivalent to show that M_i is SPD.
  By definition M_i is symmetric:

  $$\forall x \neq 0 < x, M_i x >=< x, R_i S R_i^T x >=< R_i^T x, S R_i^T x >$$

  In addition

  $$\left.\begin{array}{l} R_i \text{ is full rank} \Rightarrow R_i^T x \neq 0 \\ S \text{ is SPD} \end{array}\right\} \Rightarrow < R_i^T x, S R_i^T x > \text{ is strictly positive.}$$

- M_{loc} is SPD.
  Let $x \in U$.

  $$< x, M_{loc} x >=< x, \sum_{i=1}^{p} R_i^T M_i^{-1} R_i x >= \sum_{i=1}^{p} < R_i x, M_i^{-1} R_i x >, \tag{II.2.11}$$

  where $\forall i < R_i x, M_i^{-1} R_i x >\geq 0$ since M_i^{-1} is SPD. So the expression (II.2.11) can be zero if and only if $\forall i < R_i x, M_i^{-1} R_i x >= 0$ which implies that $x = 0$ since R_i are canonical restrictions such that $U_i = Im(R_i)$ and $U = U_2 + ... + U_p$.

■

Remark II.2.3 *If $U = U_1 \oplus .... \oplus U_n$, then M_{loc} is a block Jacobi preconditioner. Otherwise, M_{loc} is a block diagonal preconditioner with an overlap between the blocks as $U_i \cap U_j \neq \emptyset$. In this case, the preconditioner can be viewed as an algebraic additive Schwarz preconditioner for the Schur complement.*

The preconditioners are requested to be efficient on parallel distributed memory platforms. Therefore, we do mainly consider subspaces U_i that involve information mainly stored in the local memory of the processors; that is information associated with only one subdomain and its closest neighbors. We present four different decompositions of U. The first three are applicable on structured or unstructured discretizations and are defined by associating each subspace respectively with:

1. each edge E_k and each vertex v_l of the decomposition giving rise to the edge preconditioner described in Section 2.2.1,
2. each edge E_k enlarged with neighbors of its ended points v_ℓ resulting in the vertex-edge preconditioner presented in Section 2.2.2,
3. each interface Γ_i of the subdomains giving the subdomain preconditioner presented in Section 2.2.3.

The last proposed preconditioner exploited the underlying structure of the regular meshes and is similar to the alternating line relaxation on Cartesian grids. Section 2.2.4 is devoted to its description.

2.2.1 Edge preconditioners

For each edge E_i we define $R_i \equiv R_{E_i}$ as the standard pointwise restriction of nodal values on E_i. Its transpose extends grid functions in E_i by zero to the rest of the interface. Thus, $S_{ii} = R_{E_i} S R_{E_i}^T = M_i$. Similarly, we consider R_{v_l} the restriction operator for each vertex of the coarse mesh τ^H defined by the decomposition. Using the above notation we define the edge-based local preconditioner by

$$M_{loc} = M_E = \sum_{E_i} R_{E_i}^T S_{ii}^{-1} R_{E_i} + \sum_{v_l} R_{v_l}^T S_{v_l v_l}^{-1} R_{v_l}. \tag{II.2.12}$$

This preconditioner aims at capturing the interaction between neighboring nodes within the same edge interface. Notice that $S_{v_l v_l}$ in (II.2.12) is just a scalar which is the diagonal coefficient of the equation associated with the vertex v_l; this only corresponds to a diagonal scaling at the vertices of τ^H. This preconditioner is the straightforward block Jacobi that is well-known to be efficiently parallelizable. The main criticism against M_E is that it does not manage consistently neighbor nodes that are close to a vertex but belong to different edges, see Figure II.2.1. We describe in the next section a preconditioner that intends to address this deficiency.

2.2.2 Vertex-edge preconditioners

The vertex-edge preconditioner is similar to the Vertex-Space preconditioner introduced in [146], for which we merge into a single subspace the edge and vertex subspaces that appear in an additive way in [60, 146].

In Figure II.2.2, we depict U_k, the image of the restriction operator $R_k \equiv R_{VE_k}$ associated with the vertex-edge E_k. With this notation the vertex-edge preconditioner is defined by

$$M_{loc} = M_{VE} = \sum_{E_i} R_{VE_i}^T M_{VE_i}^{-1} R_{VE_i} \text{ with } M_{VE_i} = R_{VE_i} S R_{VE_i}^T.$$

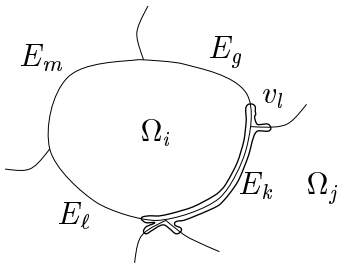


FIGURE II.2.2: U_k associated to the vertex-edge preconditioner.

In that case, two neighbor vertex-edges (for instance, E_k and E_g in Figure II.2.2) intercept each other, then the associated space splitting $(U_i)_i$ does not define a direct sum of the space U and the number of nodes in the neighborhood of the vertex v_l defines the amount of overlap between the blocks M_i of the preconditioner.

2.2.3 Subdomain preconditioner

In this alternative, we try to exploit all the information available on each subdomain and we associate each subspace U_i with the entire boundary Γ_i of subdomain Ω_i. Here, we have $R_i \equiv R_{\Gamma_i}$. Consequently $M_i = \bar{S}^{(i)}$ is the assembled local Schur complement. This splitting $(U_i)_i$ is not a direct sum of the space U and we have introduced some overlap between the blocks defining the subdomain preconditioner M_S.

We should notice the similitude between M_S and the Neumann-Neumann preconditioner, denoted M_{NN}, that was originally proposed in [25] and [52].

M_S can be written as:

$$M_S = \sum_{i=1}^{N} R_{\Gamma_i}^T (\bar{S}^{(i)})^{-1} R_{\Gamma_i} \ ,$$

while the Neumann-Neumann preconditioner is

$$M_{NN} = \sum_{i=1}^{N} R_{\Gamma_i}^T (D_i (S^{(i)})^+ D_i) R_{\Gamma_i}. \tag{II.2.13}$$

In Equation (II.2.13) the matrices D_i are weighted matrices such that $\sum_{i=1}^{N} R_{\Gamma_i}^T D_i R_{\Gamma_i} = I$. I denotes the identity matrix and $(S^{(i)})^+$ is the Moore-Penrose pseudo-inverse [94] since the local Schur complement matrices $S^{(i)}$ are singular for internal subdomains. Notice that assembling the local Schur complement $\bar{S}_i$ removes these singularities.

2.2.4 Alternating line preconditioner

The special structures of the 2D decomposition of structured meshes can be exploited to design a preconditioner similar to the alternating line relaxation. Specifically, we consider the restriction of the Schur complement operator to each grid line. In that case the definition of the restriction operator R_i differs depending on

whether the grid line is aligned with a set of domain interfaces, we denote R_k^{line} the restriction operator enabling to define M_{line_k} the restriction of the Schur complement to U_k depicted in Figure II.2.3 for a vertical interface;

or whether the grid line crosses a set of domains. In this later situation we note R_k^{across} the restriction operator enabling to define M_{line_k} the restriction of the Schur complement to U_k depicted in Figure II.2.4 for a horizontal grid line.

The resulting alternating line preconditioner can be written:

$$M_{AL} = M_{AL-x} + M_{AL-y},$$

where

$$M_{AL-x} = \sum_{k \in \left\{ \begin{array}{c} \text{horizontal grid} \\ \text{lines aligned} \\ \text{with an interface} \end{array} \right\}} (R_k^{line})^T M_{line_k}^{-1} R_k^{line} + \sum_{k \in \left\{ \begin{array}{c} \text{horizontal grid} \\ \text{lines not aligned} \\ \text{with an interface} \end{array} \right\}} (R_k^{across})^T M_{across_k}^{-1} R_k^{across},$$

and a similar definition for M_{AL-y} but for the vertical grid lines.

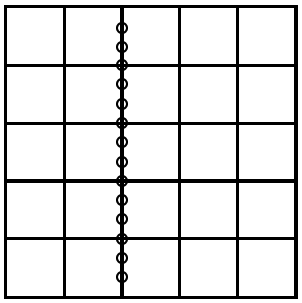


FIGURE II.2.3: U_k associated with a grid line aligned with a vertical interface.

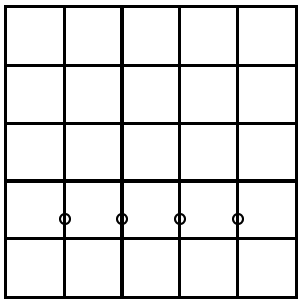


FIGURE II.2.4: U_k associated with an horizontal grid line that crosses the subdomains.

On a uniform grid with n_x grid points in the x direction partitioned among N_x equi-sized rectangles, a horizontal grid line that is aligned with the interfaces has n_x points while a line between the interfaces has $N_x - 1$ points. For a grid line aligned with an interface, M_{line_k} the restriction of the Schur complement matrix to this line is almost a block diagonal matrix (actually the matrix is block diagonal with blocks that overlap at the vertices v_l). For a grid line that crosses the subdomains, M_{across_k} the restriction of the Schur complement to this line is a tridiagonal matrix.

2.2.5 Computing alternatives

The construction of the proposed local preconditioners can be computationally expensive because the exact local Schur complement $S^{(i)}$ needs to be formed explicitly and then dense matrices M_i should be factorized. To alleviate these costs we propose several alternatives that can be combined. The first intends to reduce the construction cost of $S^{(i)}$ by using approximated solution of the local Dirichlet problems A_{ii}; the second intends to reduce the storage and the computational cost to apply the preconditioner by using sparse approximation of the M_i obtained by dropping the smallest entries. Finally the probing technique proposed by [43] for the edge preconditioner can be adapted to vertex-edge following [45] and extended to the alternating line preconditioner [89].

Local Schur with inexact local solvers

Using the up-to-date sparse direct technology of efficient sparse direct solver, A_{ii} is factorized and $S^{(i)}$ can be computed via many forward/backward substitutions. Nonetheless, this procedure remains computationally expensive. To alleviate this cost, the exact solution of the local Dirichlet problems A_{ii}^{-1} (see Equation (II.2.7)) can be replaced by some cheap approximations. For symmetric positive definite problems, approximations can be efficiently computed either by approximate inverses like AINV [17] or FSPAI [114], or by an Incomplete Cholesky factorization, ILL^T resulting in an approximate Schur complement $\tilde{S}$.

Lemma II.2.2 *If the matrix* $A = \begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{I\Gamma}^T & A_{\Gamma\Gamma} \end{pmatrix}$ *is a Stieltjes matrix and* (LL^T) *is an incomplete Cholesky factorization of* A_{II} *then* $\tilde{S} = A_{\Gamma\Gamma} - A_{I\Gamma}^T (LL^T)^{-1} A_{I\Gamma}$ *is also an Stieltjes matrix.*

Proof
It is enough to show that

$$0 \leq (LL^T)^{-1} \leq A_{II}^{-1},$$

since Theorem 7.1 in [7] will then insure that the resulting approximate Schur is a M-matrix. By construction $\tilde{S}$ is symmetric then is a Stieltjes matrix consequently SPD.
A_{II} is a symmetric M-Matrix, so by Theorem 2.4 in [123], $A_{II} = (LL^T) - R$ is a regular splitting (i.e. $(LL^T)^{-1} \geq 0$ and $R \geq 0$).

$$A_{II} = (LL^T) - R \Rightarrow (LL^T)^{-1} A_{II} = I - (LL^T)^{-1} R \leq I.$$

Since A_{II} is a M-matrix, $A_{II}^{-1} \geq 0$ then

$$0 \leq (LL^T)^{-1} \leq A_{II}^{-1}.$$

■

We note that the same property holds for approximate Schur complement computed with AINV. In [18] it is shown that the approximate inverse G of a M-matrix A computed by AINV also satisfies the inequality $0 \leq G \leq A^{-1}$.

Notice that Lemma II.2.1 and II.2.2 insure that for M-matrices the local preconditioners built using either ILL^T or AINV are SPD.

Sparse approximation of the local Schur complement

Another possible alternative to get a cheaper preconditioner is to consider a sparse approximation for S in (II.2.10) which results in a saving of memory to store the preconditioner and saving of computation to factorize and apply the preconditioner. This approximation $\hat{S}$ can be constructed by dropping the elements of S that are smaller than a given threshold. More precisely the following dropping strategy can be applied:

$$\hat{s}_{ij} = \begin{cases} 0 & \text{if } s_{ij} \leq \eta(|s_{ii}| + |s_{jj}|), \\ s_{ij} & \text{else.} \end{cases} \tag{II.2.14}$$

Lemma II.2.3 *If the matrix* $A = \begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{I\Gamma}^T & A_{\Gamma\Gamma} \end{pmatrix}$ *is a Stieltjes matrix then the sparse approximation* $\hat{S}$ *computed by* (II.2.14) *applied to* $S = A_{\Gamma\Gamma} - A_{I\Gamma}^T A_{II}^{-1} A_{I\Gamma}$ *is also an Stieltjes matrix.*

Proof
It is well-known that S is a Stieltjes matrix (see [7] for instance), then it is easy to see that removing off diagonal entries while preserving symmetry preserves this property.

■

We note that these first two alternatives can be combined, that is dropping the smallest entries of approximate M_i, to produce preconditioner cheap to compute and to store. We mention that, for M-matrices, this combination still gives rise to preconditioners that are SPD.

The probing approximation

The probing technique proposed by [43] can be followed to cheaply construct approximations for the edge and vertex-edge preconditioners. In addition this technique can also be extended to construct cheap approximation of the alternative line preconditioner, but special attention should be paid for anisotropic problems [89].

To simplify the presentation, we first consider two subdomains divided by one interface and refer to [43] for a complete analysis of this technique. The main idea is to approximate the interface matrix by a matrix having a prescribed sparsity pattern. This sparsity pattern is usually chosen as a band matrix and is motivated by the observation that the entries of S decay rapidly from the diagonal; it can be shown [93] that $|s_{ij}| = \mathcal{O}(\frac{1}{|i-j|^2})$. A banded approximation is built with matrix-vector products between S and a few carefully chosen vectors. The choice of these vectors is based on the fact that it is possible to recover the entries of an implicit band matrix having upper and lower bandwidth d from its action on $2d+1$ probe vectors defined by

$$p_k^{(j,d)} \begin{cases} 1 & if \quad k \quad mod(2d+1) \quad = \quad j \\ 0 & else \end{cases} \tag{II.2.15}$$

where the index k is used to denote the specific entries of the jth probe vector $p^{(j,d)}$. To illustrate the idea, the case $d = 1$ is considered. Specifically, it is possible to obtain all the coefficients of a tridiagonal matrix, C, using $p^{(0,1)} = (0,0,1,0,0,1,0,0,\cdots)^T$, $p^{(1,1)} = (1,0,0,1,0,0,\cdots)^T$, and $p^{(2,1)} = (0,1,0,0,1,0,0,\cdots)^T$:

$$\begin{pmatrix} c_{11} & c_{12} & & & & \\ c_{21} & c_{22} & c_{23} & & & \\ & c_{32} & c_{33} & c_{34} & & \\ & & c_{43} & c_{44} & c_{45} & \\ & & & \ddots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ \vdots & \vdots & \vdots \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & 0 \\ c_{21} & c_{22} & c_{23} \\ c_{34} & c_{32} & c_{33} \\ c_{44} & c_{45} & c_{43} \\ \vdots & \vdots & \vdots \end{pmatrix}.$$

In the two subdmain case the Schur complement is not tridiagonal but is full with entries decaying rapidly from the diagonal. In this case, the probe is not exact but gives rise to a good approximation. To generalize the probing technique to multiple domains, a band matrix must be generated for each subdomain edge. To do this, [45] defines composite probe vectors using the individual edge probes, $p^{(j,d)}$, defined by (II.2.15) for the two domain case. Specifically, they construct $2d+1$ composite vectors using the $p^{(j,d)}$'s on all the vertical edges and 0's on the horizontal edges. Then, another $2d+1$ composite vectors are constructed using the $p^{(j,d)}$'s on the horizontal edges and 0's on the vertical edges (see Figure II.2.5). We refer to this procedure as vertical/horizontal probing and use the notation $\tilde{M}_E^{vh(d)}$ to denote the local preconditioning operator.

The primary problem when probing is to avoid interference between edges that are probed simultaneously. That is, all edges which border the same subdomain influence each other. The

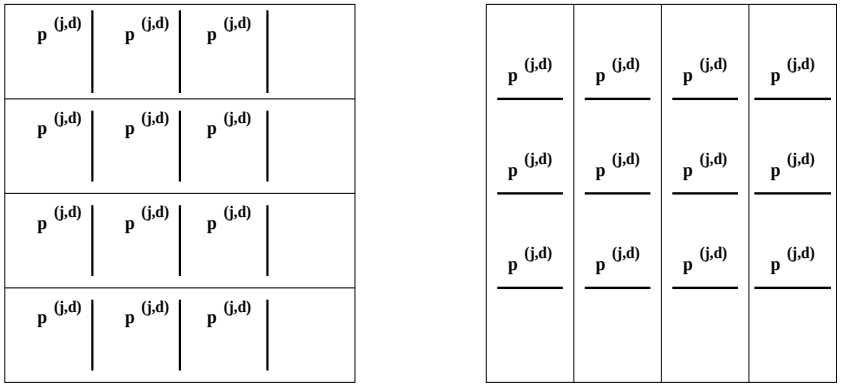


FIGURE II.2.5: Vertical/horizontal probe vectors.

vertical/horizontal probing eliminates approximation errors arising from coupling between vertical and horizontal edges. However, it does not eliminate coupling errors between the vertical edges or between the horizontal edges. In the case of isotropic problems, this coupling is usually much weaker than the coupling within an edge and so the vertical/horizontal probing is sufficient. However, for 2D anisotropic problem as

$$\varepsilon \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f \text{ with } \varepsilon \ll 1, \tag{II.2.16}$$

there exists a sufficiently small ε such that the coupling between two horizontal edges that border the same subdomain is actually stronger than the coupling within the horizontal edges. In this case, the use of the composite 'horizontal' probe vectors defined in Figure II.2.5 (used to construct approximations to coupling within each horizontal edge) will give rise to inaccurate approximations of the associated S_{ii} due to the large coupling between these horizontal edges.

To alleviate this interference problem we propose to further subdivide both horizontal and vertical edges into 'red' and 'black' sets to minimize the approximation errors arising from coupling between vertical (or horizontal) interfaces (see Figure II.2.6 for the vertical interfaces). We refer to

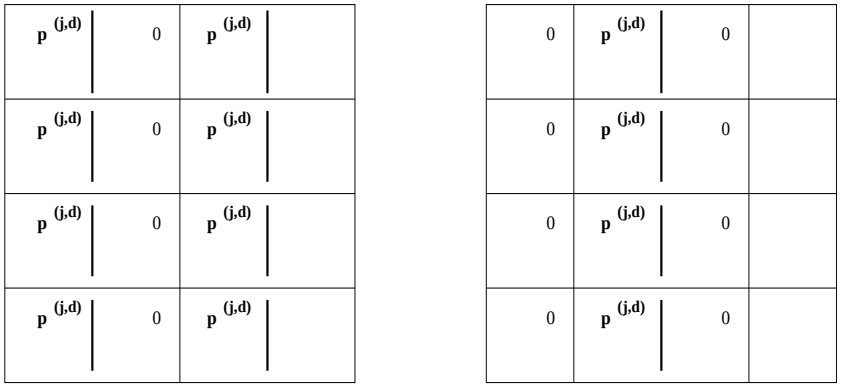


FIGURE II.2.6: Red/black vertical probe vectors.

this procedure as red/black probing and use the notation $\tilde{M}_E^{rb(d)}$ to denote the local component of the preconditioner. The important aspect of red/black probing is that the probe approximation on

a particular edge is not distorted by other edges as there is no coupling in the Schur complement matrix between edges within the same line and between two edges on parallel non-neighboring interfaces. We note that while this procedure is for regular grids, nonuniform grids can use the same idea via multicoloring. Essentially, two edges belonging to the same subdomain must be assigned to different colors. To summarize, the overall costs of building $\tilde{M}_E^{vh(d)}$ and $\tilde{M}_E^{rb(d)}$ are $2(2d+1)$ and $4(2d+1)$ matrix-vector products respectively. For $d = 1$, this corresponds to 6 and 12 matrix-vector products. Though the vertical/horizontal probing is the least expensive, the difference in the convergence rate is often large enough for anisotropic problems to offset the extra cost.

The probing technique can also be adapted to construct cheap approximation of the submatrices of the Schur complement involved in the alternating line preconditioner M_{AL}. The construction of this preconditioner consists of probing the Schur complement along the following:

1. vertical grid lines aligned with interfaces using the $p^{(j,d)}$ probe vectors over each entire interface as shown in Figure II.2.7 in the case of vertical grid lines aligned with vertical interfaces,
2. horizontal grid lines not aligned with interfaces using probe vectors, $q^{(i,1)}$, defined over each element of the interfaces that lies on that line. Such a vector is depicted in Figure II.2.8 for a horizontal grid line,
3. horizontal grid lines aligned with interfaces, and
4. vertical grid lines not aligned with interfaces.

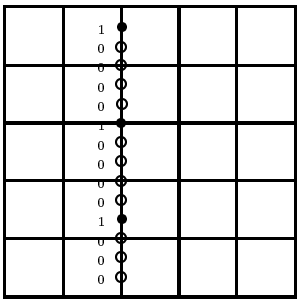


FIGURE II.2.7: Example of a $p^{(4,2)}$ vector.

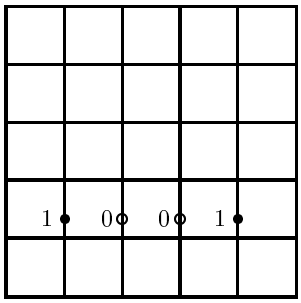


FIGURE II.2.8: Example of a $q^{(1,1)}$ vector.

To reduce the cost of the building of these matrices, steps 1) and 2) can be combined as well as steps 3) and 4). For example, for steps 1) and 2) the probe vectors on all the vertical interfaces are defined by

$$q^{(i,1)} \otimes p^{(j,d)} \quad \begin{array}{l} i = 0, ..., 2 \\ j = 0, ..., 2d \, ; \end{array}$$

we recall that $(A \otimes B)_{km} = a_{km}B$. This scheme is depicted in Figure II.2.9 for the case $d = 2$ for points on vertical interfaces. After steps 1) and 2) we repeat the same procedure for the horizontal interface points. It is important to note that combining steps 1) and 2) introduces small approximation errors. For example, the tridiagonals constructed for lines that are not aligned with interfaces are no longer exact. However, the advantage of this process is that only $6(2d+1)$ matrix-vector products are needed to build the probe (as compared with $4(2d+1)$ for red/black probing). We refer to this procedure as alternating line probing and use the notation $\tilde{M}_{AL}^{(d)}$ to denote the associated local preconditioning operator.

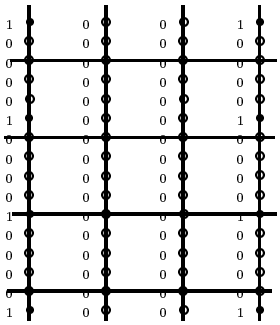


FIGURE II.2.9: Sample of $(q^{(i,1)} \otimes p^{(j,2)})$.

It can be shown (see for instance [44]) that the local preconditioners alone are not numerically scalable for elliptic problems in the sense that

$$\kappa(M_{loc}S) = \mathcal{O}(H^{-2}). \tag{II.2.17}$$

This means that when the number of subdomains increases the number of conjugate gradient iterations increases as well. To ensure the numerical scalability of the preconditioners a quasi-optimality property should be satisfied; that is, the condition number of the associated preconditioned systems is independent from the number of subdomains and only logarithmically dependent on the size of the subdomains. In this respect, a coarse problem defined on the whole physical domain must be incorporated into the preconditioner.

2.3 Coarse space preconditioners

Various domain decomposition techniques, from the eighties and nineties, have suggested different global coupling mechanisms and various combinations between them and the local preconditioners. In this section, we present a contribution in the area of two-level domain decomposition methods using algebraic constructions of the coarse space for solving heterogeneous, anisotropic two-dimensional elliptic problems on structured or unstructured discretizations. They are closely related to BPS [26], although we propose different coarse spaces to construct their coarse components.

The classical BPS preconditioner [26] can be briefly described as follows. Let assume that Ω_1, ..., Ω_N form the elements of a coarse grid mesh, τ^H, with mesh size H. We then define grid transfer operators between the interface and the coarse grid. R^T is an interpolation operator which corresponds to using linear interpolation between two adjacent cross points v_j, v_k (i.e. adjacent points in τ^H connected by an edge E_i) to define values on the edge E_i. Finally, A_H is the Galerkin coarse grid operator $A_H = RAR^T$ defined on τ^H.

With these notations a very close variant of the BPS preconditioner is defined by

$$M_{BPS} = \sum_{E_i} R_{E_i}^T S_{ii}^{-1} R_{E_i} + R^T A_H^{-1} R, \tag{II.2.18}$$

as described, for instance, in this algebraic form in [44]. It can be interpreted as a generalized block Jacobi preconditioner for the Schur complement system (II.2.5) where the block diagonal preconditioning for S_V is omitted and a residual correction is used on a coarse grid. The coarse

grid term $R^T A_H^{-1} R$ allows to incorporate a global coupling between the interfaces, which enables to get the condition number given in Equation II.2.1 that is independent of the number of subdomains.

In the rest of this section, we will define various preconditioners that only differ in the definition of M_{global} that appears in (II.2.2). Our goal is to obtain performance similar to those of the original BPS preconditioner even in presence of anisotropy or heterogeneity, with a simple algebraic structure and a parallel implementation strategy. In particular, for practical implementation purposes within a general purpose computer code, we do not want to refer explicitly to an underlying coarse grid, or to underlying basis functions, since these notions are always hard to identify in practice when using general grids, finite elements or mixed finite elements.

Let U be the algebraic space of nodal vectors where the Schur complement matrix is defined and U_0 be a q-dimensional subspace of U. Elements of U_0 are characterized by the set of nodal values that they can achieve that will be referred to as the support of the vectors. This subspace will be called coarse space. Let $R_0 : U \to U_0$ be a restriction operator which maps full vectors of U into vectors in U_0, and let $R_0^T : U_0 \to U$ be the transpose of R_0, an extension operator which extends vectors from the coarse space U_0 to full vectors in the fine space U. The Galerkin coarse space operator

$$A_0 = R_0 S R_0^T, \tag{II.2.19}$$

in some way, represents the Schur complement on the coarse space U_0. The global coupling mechanism is introduced by the coarse component of the preconditioner which can thus be defined as $M_{global} = R_0^T A_0^{-1} R_0$.

The following lemma ensures the correctness of the operators we work with:

Lemma II.2.4 *If the operator R_0^T is of full rank and if S is non-singular, symmetric and positive definite, then the matrix A_0, defined in Equation (II.2.19), is non-singular, symmetric and positive definite.*

The coarse-space preconditioners will only differ in the choice of the coarse space U_0 and the interpolation operator R_0^T. For convergence reasons, and similarly to the Neumann-Neumann and Balancing Neumann-Neumann preconditioner [116, 118], R_0^T must be a partition of the unity in U in the sense that

$$R_0^T \mathbf{1} = \mathbf{1}, \tag{II.2.20}$$

where the symbol $\mathbf{1}$ denotes the vectors of all 1's that have different size in the right and left hand side of (II.2.20).

With these notations and definitions, all the preconditioners presented in the sequel of this section can be written as follows:

$$M = \sum_{E_i} R_i^T \tilde{S}_{ii}^{-1} R_i + R_0^T A_0^{-1} R_0, \tag{II.2.21}$$

where we usually replace S_{ii} in (II.2.18) by an approximation $\tilde{S}_{ii}$ computed using a multicolouring probing technique.

In the next section, we define various coarse spaces and restrictions operators which can be used in a very general framework. The support of the basis vectors Z_k has inspired the name of the coarse spaces. Although a large number of different preconditioners can then be proposed, we restrict our study to five combinations of coarse spaces and restriction operators.

2.3.1 Vertex based coarse space

The first coarse space we consider is similar to the one of BPS. Each degree of freedom of U_0 is associated with one vertex v_j, and the basis vectors generating the nodal values of the elements of U_0 can be defined as follows. Let $\{v_k\} \subset V$ be a singleton set that contains the index associated with a cross point and (E_j) be the adjacent edges to v_k. Let m_c denote the number of vertex points then

$$\widetilde{\mathcal{I}}_k = \bigcup_j E_j \cup \{v_k\}$$

is the set of indices we associate with the cross point v_k to define the support of the basis vectors.

Let Z_k be a vector defined on B and $Z_k(i)$ its i-th component. Then, the vertex based coarse space U_0 can be defined as

$$U_0 = \text{span}[Z_k : k = 1, \ldots, m_c], \text{ where } \quad Z_k(i) = \begin{cases} 1 & \text{if } i \in \widetilde{\mathcal{I}}_k, \\ 0 & \text{elsewhere.} \end{cases}$$

The set $\widetilde{\mathcal{I}}_k$ associated with a cross point V_k is depicted in Figure II.2.10. The set of vectors $\mathcal{B} = \{Z_1, Z_2, \ldots, Z_{m_c}\}$ forms a basis for the subspace U_0, as these vectors span U_0 by construction and they are linearly independent.

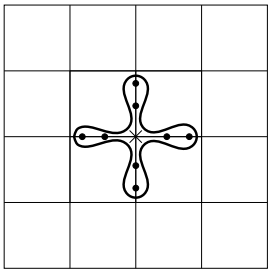


FIGURE II.2.10: Support of one basis vector of the "vertex" coarse space.

For this coarse space, we consider three different restriction operators R_0 and their associated prolongation operator R_0^T.

Flat restriction operator

This operator returns for each set $\widetilde{\mathcal{I}}_k$ the weighted sum of values of all the nodes in $\widetilde{\mathcal{I}}_k$. The weights are determined by the inverse of the number of sets $\widetilde{\mathcal{I}}_k$, $k \in \{1, 2, \ldots, m_c\}$, that a given node belongs to. For 2D problem discretized by finite differences, the weight for the cross points is 1 and for an edge point is 1/2.

Linear interpolation operator

The interpolation operators in this section and the next one are basically 1D interpolations on the edges E_i of values defined at the cross points that are its end-points. In this respect, let us describe them in the 1D framework obtained by mapping the edge E_i on the interval (0,1) through

the map $\phi(P_j) = \frac{\sum_{k\leq j} |P_{k-1}P_k|}{\sum_{k\in E_i} |P_{k-1}P_k|}$. We consider the following 1D model problem

$$\left\{ \begin{array}{rcl} -\frac{d}{dx}(a(x)\frac{d}{dx}u(x)) & = & f \quad \text{in (0.1)}, \\ u(x) & = & 0 \quad \text{at } x\text{=0 and 1}. \end{array} \right. \tag{II.2.22}$$

Let $H^1(0,1)$ be the standard Sobolev space on the interval (0,1) and $H_0^1(0,1)$ its subspace whose functions vanish at $x = 0$ and $x = 1$. Given a grid $x_j^h = jh, j = 0, ..., n$ on (0,1) as the image of the original discretization of E_i, define the fine grid linear finite element space to be

$$V^h = \{v^h \in H_0^1(0,1) : v^h \text{ is linear on } [x_j^h, x_{j+1}^h], j = 0, ..., n-1\},$$

and denote the set of nodal basis by $\{\phi_j^h\}_{j=0}^n$.

Let $(x_i^H)_{i=1,m}$ be the set of coarse grid points defined by the vertices of the partitioning of $(0,1)$ into non-overlapping subdomains. Now, we define the coarse subspace $V^H = span\{\phi_i^H : i = 1, ..., m\}$ where ϕ_i^H are the coarse grid nodal basis functions.

Since $\{\phi_i^H\}$ is a basis of V^H, which is a subspace of V^h, there exists a unique matrix R_0^T of size $n-1 \times m$ such that

$$[\phi_1^H ... \phi_m^H] = [\phi_1^h ... \phi_{n-1}^h] R_0^T,$$

that is usually referred to as the local interpolation matrix. We depict in Figure II.2.11 an example of a coarse grid basis functions that defines the linear interpolation.

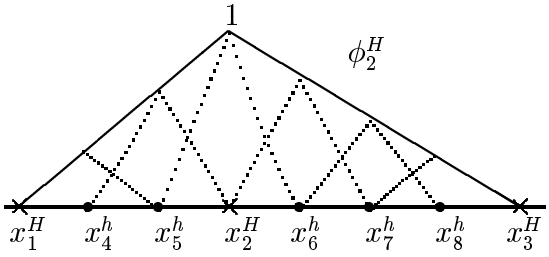


FIGURE II.2.11: coarse grid basis function ϕ_2^H associated with the linear interpolation.

When using block Jacobi as the local preconditioner combined with the vertex coarse space using this linear prolongation operator, the resulting preconditioner is equivalent to the genuine BPS [26].

Let A be the matrix defined in Equation (II.2.4). Let A_V be the Galerkin coarse grid operator associated with of A defined by

$$A_V = \widetilde{R}_0 A \widetilde{R}_0^T, \tag{II.2.23}$$

where $\widetilde{R}_0$ is a restriction operator from Ω to the coarse space U_0. It has been shown [26, 147] that in general for elliptic problems the operator A_V is spectrally equivalent to

$$R_0 S R_0^T \tag{II.2.24}$$

and for a few cases these coarse operators are even equal.

If we have used the approach defined by (II.2.23), the construction of the coarse space would have been reduced to some matrix-vector multiplications with A, and the factorization of A_V. Nevertheless, we deal with problems for which only the spectral equivalence between (II.2.23) and (II.2.24) is ensured. For this reason, we have preferred to use the Galerkin coarse grid correction with the Schur complement matrix as described in (II.2.24) rather than the one proposed in a similar matrix form in [44] that used A_V defined by (II.2.23).

Operator-dependent restriction operator

The origin of the operator-dependent restriction is the operator-dependent transfer operator proposed in the framework of multigrid methods, see [156] and references therein. In [87], we proposed an extension of these operators for two-dimensional non-overlapping domain decomposition methods. The general purpose of these operators is to construct from u defined on V an interpolation $\tilde{u}$ defined on B that is piecewise linear so that $a\frac{\partial \tilde{u}}{\partial x}$ and $b\frac{\partial \tilde{u}}{\partial y}$ are continuous even when either a or b in (II.2.3) are discontinuous along an edge E_i.

Similarly to the linear interpolation, we can define the operator-dependent interpolation in 1D through the definition of the coarse grid basis functions ϕ_i^H. In that case those basis functions are constructed by solving the following local problem in $[x_{i-1}^H, x_i^H]$:

$$\begin{cases} -\dfrac{d}{dx}(a(x)\dfrac{d}{dx}\phi_i^H) = 0 \quad \text{in } [x_{i-1}^H, x_i^H], \\ \phi_i^H(x_{i-1}^H) = 0, \phi_i^H(x_i^H) = 1. \end{cases} \tag{II.2.25}$$

In Figure II.2.12, we depict the basis function ϕ_2^H when the function $a(x)$ is piecewise constant with some discontinuities at x_5^h and x_7^h.

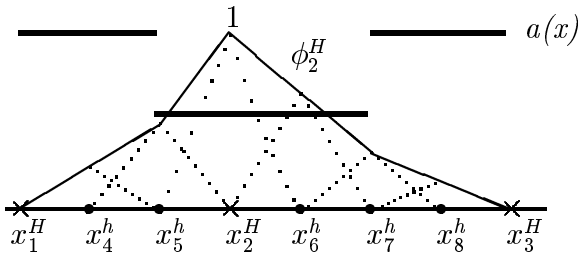


FIGURE II.2.12: coarse grid basis function ϕ_2^H associated with the operator-dependent interpolation.

We omit the computational details, and only notice that such operators

- can be constructed by solving one tridiagonal linear system for each E_i, that corresponds to the solution of (II.2.25). The size of the tridiagonal matrices is the number of nodes on E_i,
- reduce to a linear interpolation when $a() = 1$ and $b() = 1$,
- in 1D reduce to the multigrid energy minimization approach [154, 155] or to the multigrid operator-dependent interpolation with harmonic averaging [87], when every other point is a coarse point (i.e. each subdomain contains only one point).

2.3.2 Subdomain based coarse space

With this coarse space, we associate one degree of freedom with each subdomain. Let B be as defined in Equation (II.2.6). Let Ω_k be a subdomain and $\partial\Omega_k$ its boundary. Then

$$\overline{\mathcal{I}}_k = \partial\Omega_k \cap B$$

is the set of indices we associate with the domain Ω_k. Figure II.2.13 shows the elements of a certain set $\overline{\mathcal{I}}_k$.

Let Z_k a vector defined on B and $Z_k(i)$ its i-th component. Then, the subdomain-based coarse space U_0 can be defined as

$$U_0 = \text{span}[Z_k : k = 1, \dots, N], \text{ where } Z_k(i) = \begin{cases} 1, & \text{if } i \in \overline{\mathcal{I}}_k \text{and} \\ 0, & \text{otherwise.} \end{cases}$$

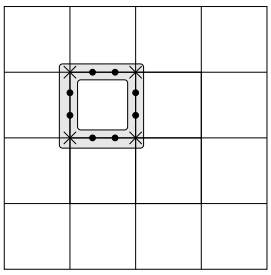


FIGURE II.2.13: Support of one basis vector of the "subdomain" coarse space.

Notice that for the example depicted in Figure II.2.13, $[Z_k]$ is rank deficient. Indeed, if we consider $\tilde{v} = \sum_{i=1}^{N} \alpha_i Z_i$ where the α_i are , in a checker-board pattern, equal to -1 and $+1$, it is easy to see that $\tilde{v} = 0$. Nevertheless, this rank deficiency can be easily removed by discarding one of the vectors of $[Z_k]$. In this particular situation, the set of vectors $\mathcal{B} = \{Z_1, Z_2, \dots, Z_{N-1}\}$ forms a basis for the subspace U_0.
The considered restriction operator R_0 returns for each subdomain $(\Omega_i)_{i=1,N-1}$ the weighted sum of the values at all the nodes on the boundary of that subdomain. The weights are determined by the inverse of the number of subdomains in $(\Omega_i)_{i=1,N-1}$ each node belongs to. For all the nodes but the ones on $\partial\Omega_N$ (in our particular example) this weight is: $1/2$ for the points on an edge and $1/4$ for the cross points. These weights can be replaced as in [116] by operator dependent weights $R_0(i,k) = a_i/(a_i + a_j)$ on the edge separating Ω_i from Ω_j , but this choice has not been tested numerically in the present work.

Remark II.2.4 *Although used in a completely different context, this coarse space is similar to the one used in the Balancing Neumann-Neumann preconditioner for Poisson-type problems [118]. We use one basis vector for each subdomain, whereas in Balancing Neumann-Neumann the basis vectors are only defined for interior subdomains for solving the Dirichlet problem* (II.2.3)*, that are the subdomains where the local Neumann problems are singular.*

2.3.3 Edge based coarse space

We refine the coarse space based on the subdomains and we introduce one degree of freedom per interface between two neighboring subdomains, that is, when $\partial\Omega_i$ and $\partial\Omega_j$ share at least one edge of the mesh.

Let E_k be an edge and v_j and v_l its adjacent cross points then

$$\hat{\mathcal{I}}_k = E_k \cup \{v_j\} \cup \{v_l\}$$

is the set of indices we associate with the edge E_k.

Let Z_k defined on B and $Z_k(i)$ its i-th component. Let m_e denote the number of edges $E_i \subset B$, then, the edge based coarse space U_0 can be defined as:

$$U_0 = \text{span}[Z_k : k = 1, \ldots, m_e], \text{ where } \quad Z_k(i) = \begin{cases} 1 & i \in \hat{\mathcal{I}}_k, \\ 0 & \text{otherwise.} \end{cases}$$

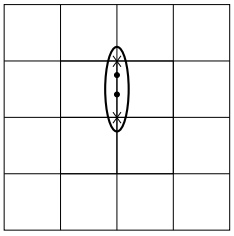


FIGURE II.2.14: Support of one basis vector of the "edge" coarse space.

The set $\hat{\mathcal{I}}_k$ associated with an element of the coarse space U_0 is depicted in Figure II.2.14. The set of vectors $\mathcal{B} = \{Z_1, Z_2, \ldots, Z_{m_e}\}$ forms a basis for the subspace U_0, as before, these vectors span U_0 by construction and they are linearly independent.

The considered restriction operator R_0 returns for each edge the weighted sum of the values at all the nodes on that edge. The weights are determined by the inverse of the number of edges each node belongs to. For 2D problems discretized by finite differences the weights for the restriction operator are 1 for points belonging to an edge and one fourth for the internal cross points.

2.4 Numerical experiments

We consider the solution of Equation (II.2.3) discretized by a five-point central finite difference scheme on a uniform mesh using a preconditioned Schur complement method. The background of our study is the numerical solution of the 2D drift-diffusion equations for the simulation of semi-conductor devices [88]. In this respect, we intend to evaluate the sensitivity of the preconditioners to anisotropy and to discontinuity. We illustrate the numerical scalability of the proposed preconditioners on academic two-dimensional model test cases that have both anisotropy and discontinuity.

2.4.1 Local preconditioners

Model problems

In Figure II.2.15, we represent the unit square divided into five regions where piecewise constant functions are used to define a first set of test problems. In addition, we have performed experiments with the problem defined by piecewise constant functions as depicted in Figure II.2.16. Let $a()$ and $b()$ be the diffusion coefficients of the elliptic problem as described in Equation (II.2.3).

Using this notation and Figure II.2.15, we define the first set of model problems with different degrees of difficulty:

- Poisson problem: $a() = 1$ and $b() = 1$,
- anisotropic and discontinuous problems with $a() = 1$ and $b() = c, d$ or f which depend on x and y.
  - AD-F1: c=1, d= 10^2 and f= 10^{-2}.

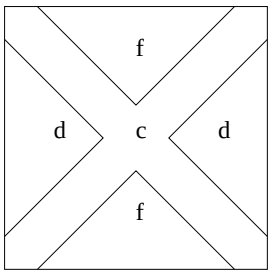


FIGURE II.2.15: Example 1 - Flag.

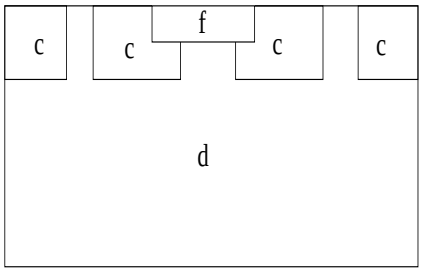


FIGURE II.2.16: Example 2 - Region.

  - AD-F2: c=1, d= 10^3 and f= 10^{-3}.

- discontinuous problems with $a() = b() = c, d, f$.
  - D-F1: c=1, d= 10^2 and f= 10^{-2}.
  - D-F2: c=1, d= 10^3 and f= 10^{-3}.

Using piecewise constant functions on the regions depicted in Figure II.2.16, we define a second set of test problems:

- anisotropic and discontinuous problems: $a() = 1$ and $b() = c, d$ or f.
  - AD-R: c= 10^1 , d= 10^{-2} and f= 10^{-1}.
- discontinuous problems: $a() = b() = c, d, f$.
  - D-R: c= 10^1 , d= 10^{-2} and f= 10^{-1}.

We have also considered a last set of problems associated with (II.2.3). We have introduced anisotropy not necessarily aligned with the axis but making an angle θ with the x-direction. For $\theta = 0$, this corresponds to the classical model anisotropic problem defined by Equation (II.2.16).

Experimental results

For the experimental results related to M_{VE}, we have considered two extra edge points in the neighborhood of the vertices v_l in each direction. For a more detailed study about the influence of the size of the overlap in the neighborhood of the vertices v_l on the convergence rate, we refer to [36] and [37]. We just state here that a very small overlap is usually enough to improve the behavior of M_{VE} with respect to M_E.

For all the experimental results reported in the next section, the convergence of the preconditioned conjugate gradient method is attained when the 2-norm of the residual of the current iteration normalized by the 2-norm of the right hand side is less than 10^{-6}, the initial guess x_0 for the conjugate gradient iterations was the null vector. All the experiments were performed in double precision arithmetic. When results with a two-level preconditioner are reported the coarse component is vertex operator dependent. The resulting variants of the BPS preconditioner will be denoted:

- M_{BPS-E} for $M_{loc} = M_E$. Notice that this local preconditioner is the one used in the genuine BPS; in this respect M_{BPS-E} is the closest variant to regular BPS. It is a slight improvement of regular BPS as the coarse component does not rely on the spectral equivalence property between A and S for uniformly elliptic operators.

- M_{BPS-VE} for $M_{loc} = M_{VE}$.
- M_{BPS-S} for $M_{loc} = M_S$.
- M_{BPS-AL} for $M_{loc} = M_{AL}$.

For comparison purpose we also consider the balanced Neumann-Neumann preconditioner that has proven to be an efficient domain decomposition preconditioner for some fairly difficult problems [116], such as linear systems arising from structural analysis. In the sequel this preconditioner will be denoted M_{BNN}.

To study the behavior of the preconditioners, we fix the ratio H/h that appears in Equation (II.2.1) while varying the number of subdomains. For all the experiments reported in the following tables, the number of subdomains varies from 16 (4×4 decomposition) up-to 256 (16×16 decomposition) keeping the size of each subdomain constant (i.e. 16×16 mesh for each subdomain, that is $\frac{H}{h} = 16$).

In Table II.2.1, we report results observed on the Poisson equation using the preconditioners with and without the coarse space component. When only local preconditioners are implemented, it can be seen that when the local information is more represented in the preconditioner, the convergence is better. These results also show that without a coarse space component the number of iterations required by the preconditioned conjugate gradient grows with the number of subdomains as predicted by the estimated condition number given by Equation (II.2.17). Using the two-level preconditioners, these observations are no longer true. The coarse space component somehow smoothes the effect of the local component. Accordingly to the theoretical bound given by Equation (II.2.1), the number of preconditioned conjugate gradient iterations becomes independent of the number of domains. Finally, we note that for the two-level preconditioners M_{BPS-E} and M_{BNN}, the results are similar to those of other authors [45, 118].

# subdomains	4×4	8×8	16×16
M_E	13	28	51
M_{VE}	12	22	40
M_S	11	19	32
M_{BPS-E}	9	11	11
M_{BPS-VE}	10	12	12
M_{BPS-S}	10	10	11
M_{BNN}	11	12	12

TABLE II.2.1: Number of iterations on the Poisson problem.

In Table II.2.2, we depict the numerical behavior of the preconditioners on the model problems that only exhibit anisotropy not aligned with the axes. When no coarse space component is implemented M_{VE} still outperforms M_E; M_S is the most efficient and the number of iterations of all the preconditioners grows with the number of subdomains. For the two-level preconditioners, we first observe that the anisotropy prevents them to have an optimal convergence behavior independent of the number of subdomains, even though the number of iterations is quite decreased by the coarse space component. Furthermore, for some problems M_{BPS-VE} becomes less efficient than the simpler M_{BPS-E} while M_{BPS-S} always ensures the fastest convergence. So the conjecture, "the richer the local preconditioner, the more efficient the preconditioner", is only true when the local preconditioners run alone.

In Tables II.2.3 and II.2.4, we study the numerical behavior of the two-level preconditioners on model problems arising from the discretization of (II.2.3) that exhibit either discontinuity

# subdomains	4×4			8×8			16×16		
θ	0	$\pi/8$	$\pi/4$	0	$\pi/8$	$\pi/4$	0	$\pi/8$	$\pi/4$
M_E	21	34	30	47	67	77	88	132	164
M_{VE}	21	22	23	44	42	59	72	81	141
M_S	14	20	20	25	40	41	53	75	88
M_{BPS-E}	27	24	20	58	34	28	81	43	35
M_{BPS-VE}	25	21	21	48	33	35	85	43	49
M_{BPS-S}	20	19	17	33	26	21	47	33	26

TABLE II.2.2: Number of iterations - Anisotropy ($\varepsilon = 10^{-3}$) with several angles.

(Table II.2.3) or both discontinuity and anisotropy (Table II.2.4). For the problems with only discontinuity, all the variants M_{BPS-*} have comparable convergence behaviors.

# subdom.	4×4			8×8			16×16		
	D-F1	D-F2	D-R	D-F1	D-F2	D-R	D-F1	D-F2	D-R
M_{BPS-E}	12	11	10	11	11	11	14	15	11
M_{BPS-VE}	13	12	11	13	12	12	16	16	12
M_{BPS-S}	12	10	10	11	11	11	14	14	11
M_{BNN}	25	27	21	29	28	38	48	65	52

TABLE II.2.3: Number of iterations for problems with discontinuity.

As it can be seen in Table II.2.4, problems with anisotropy and discontinuity are more difficult to solve. Again M_{BPS-VE} does not outperform the basic M_{BPS-E}. For those examples, similarly to the pure anisotropic situation reported in Table II.2.2, M_{BPS-S} exhibits once again the best convergence behavior.

# subdom.	4×4			8×8			16×16		
	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R
M_{BPS-E}	18	24	25	29	65	35	42	103	39
M_{BPS-VE}	18	23	24	33	80	40	56	141	55
M_{BPS-S}	16	20	18	22	43	22	33	79	26
M_{BNN}	37	52	147	60	158	644	97	311	*[1]

TABLE II.2.4: Number iterations for problems with discontinuity and anisotropy.

The relative poor performance of M_{BNN}, reported in Table II.2.3 and II.2.4, could be improved. An alternative way, as suggested in [52, 116], should be a better choice of the weight matrices D_i, involved in Equation (II.2.13), when the diagonal entries of S are available. With this appropriated choice of the weights, it can be expected a reduction of the gap between M_{BNN} and M_{BPS-*} for discontinuous problems, as suggested by the results reported in [119]. In Table II.2.5 we report the numerical behavior of M_{BPS-S} and M_{BNN} for the anisotropic problems defined by Equation (II.2.16) for different values of the anisotropic coefficient ε is varied. For those problems, the choice of the weighted matrices used in [119] for M_{BNN} would reduce to the simple ones we

[1] '*' means no convergence after 1000 iterations.

ε	1.0	10^{-1}	10^{-2}	10^{-3}
M_{BNN}	12	20	40	98
M_{BPS-S}	12	15	22	33

TABLE II.2.5: Number of iterations varying the anisotropy with a 8×8 subdomain decomposition.

have considered; that is, $\frac{1}{2}$ for the nodes on the edges and $\frac{1}{4}$ for the vertex points v_l. For anisotropic problems, we cannot expect M_{BNN} to become competitive with M_{BPS-S} for ε lower than 10^{-1}.

Local Schur with inexact local solvers To alleviate the cost of the preconditioners construction, the factorization of the local Dirichlet problem can be replaced by an incomplete Cholesky factorization without fill-in, i.e. $ILL^T(0)$, or with some fill-in controlled through a threshold, i.e. $ILL^T(t)$ [141]. In this later situation the amount of fill-in can be defined by the fill-in ratio that is the number of non-zeros in the incomplete factors divided by the number of non-zeros in the lower part of the original matrices; by definition this fill-in ratio is equal to one for $ILL^T(0)$.

In Table II.2.6 and II.2.7, we denote by $\tilde{M}_{BPS-E}$, $\tilde{M}_{BPS-VE}$ and $\tilde{M}_{BPS-S}$ the preconditioners computed using those inexact local solves. More precisely, we report in Table II.2.6 the number of iterations when $ILL^T(0)$ is used and in Table II.2.7 those observed when some fill-in is enabled with a fill-in ratio lower than 3.5.

# subdom.	4×4			8×8			16×16		
	D-F1	D-F2	D-R	D-F1	D-F2	D-R	D-F1	D-F2	D-R
$\tilde{M}_{BPS-E}$	14	13	14	13	13	14	17	17	14
$\tilde{M}_{BPS-VE}$	20	18	20	19	19	19	24	26	20
$\tilde{M}_{BPS-S}$	14	13	15	13	13	12	17	18	13
	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R
$\tilde{M}_{BPS-E}$	24	30	28	36	67	37	50	112	45
$\tilde{M}_{BPS-VE}$	27	34	31	40	84	48	64	143	64
$\tilde{M}_{BPS-S}$	24	26	23	30	53	31	47	80	41

TABLE II.2.6: Number of iterations using inexact local solvers $ILL^T(0)$ to build the preconditioners.

The comparison of the results given in Table II.2.6 and II.2.7 and those in Tables II.2.3 and II.2.4 shows that, as it could have been expected, the approximation of the local Schur complement used to build the preconditioners generally deteriorates the numerical behaviors of the preconditioner. This approximation does not affect significantly the numerical behavior of $\tilde{M}_{BPS-E}$ and $\tilde{M}_{BPS-S}$ but deteriorates noticeably the one of $\tilde{M}_{BPS-VE}$. In addition, enabling some fill-in in the incomplete factorizations generally improves the convergence rate; the most significant improvements are observed on anisotropic and discontinuous problems with $\tilde{M}_{BPS-VE}$ and $\tilde{M}_{BPS-S}$.

Sparse approximation of the Schur complement In Table II.2.8 we report the number of iterations using an approximate Schur complement $\hat{S}$ with η in (II.2.14) such that we only retains around 5 % of the entries in S. The resulting preconditioners are denoted respectively by $\hat{M}_{BPS-E}$, $\hat{M}_{BPS-VE}$ and $\hat{M}_{BPS-S}$.

# subdom.	4×4			8×8			16×16		
	D-F1	D-F2	D-R	D-F1	D-F2	D-R	D-F1	D-F2	D-R
$\tilde{M}_{BPS-E}$	13	12	12	13	14	12	16	19	11
$\tilde{M}_{BPS-VE}$	15	17	12	17	19	13	22	27	12
$\tilde{M}_{BPS-S}$	12	12	10	11	12	11	16	18	11
	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R
$\tilde{M}_{BPS-E}$	23	31	28	35	70	37	48	114	40
$\tilde{M}_{BPS-VE}$	21	33	26	36	85	44	59	147	56
$\tilde{M}_{BPS-S}$	19	27	20	27	54	24	39	81	29

TABLE II.2.7: Number of iterations using inexact local solvers $ILL^T(t)$ to build the preconditioners.

# subdom.	4×4			8×8			16×16		
	D-F1	D-F2	D-R	D-F1	D-F2	D-R	D-F1	D-F2	D-R
$\hat{M}_{BPS-E}$	13	12	12	11	11	13	14	15	12
$\hat{M}_{BPS-VE}$	16	16	18	16	16	18	20	20	18
$\hat{M}_{BPS-S}$	12	11	12	12	12	11	15	16	11
	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R
$\hat{M}_{BPS-E}$	18	25	27	29	65	35	43	111	40
$\hat{M}_{BPS-VE}$	24	30	26	36	81	42	57	142	57
$\hat{M}_{BPS-S}$	18	21	18	23	44	23	36	79	27

TABLE II.2.8: Number of iterations using sparse Schur to build the preconditioners.

The comparison of these results with those displayed in Table II.2.3 and II.2.4 indicates that, except for $\hat{M}_{BPS-VE}$ on discontinuous problems, only retaining very few entries in the Schur complement is enough to ensure the numerical quality of these preconditioners since the number of iterations are roughly the same in both cases (except for $\hat{M}_{BPS-VE}$ on discontinuous problems).

In addition, as mentioned in Section 2.2.5, the inexact local solvers and dropping strategies can be combined to build variants of the preconditioners. The resulting preconditioners are respectively denoted by $\underline{M}_{BPS-E}$, $\underline{M}_{BPS-VE}$ and $\underline{M}_{BPS-S}$. Numerical experiments where we dropped the smallest entries of the local preconditioners built using $ILL^T(t)$ are reported in Table II.2.9. Comparing these results with those of Tables II.2.8 and II.2.7 indicates that the numerical quality of the resulting preconditioners are mainly governed by the use ability of ILL^T to well approximate the local Dirichlet problems.

Probing techniques In this section we only report numerical experiments conduced to construct approximation to M_E and M_{AL} using the probing technique. The probing idea can also be applied to build effective approximations to M_{VE}, we refer to [35] and [37] were experiments are reported. The resulting two-level preconditioner are denoted

- $\tilde{M}^{vh(d)}_{BPS-E}$ when $M_{loc} = \tilde{M}^{vh(d)}$,
- $\tilde{M}^{rb(d)}_{BPS-E}$ when $M_{loc} = \tilde{M}^{rb(d)}$,
- $\tilde{M}^{(d)}_{BPS-AL}$ when $M_{loc} = \tilde{M}^{(d)}_{AL}$.

# subdom.	4×4			8×8			16×16		
	D-F1	D-F2	D-R	D-F1	D-F2	D-R	D-F1	D-F2	D-R
$\underline{M}_{BPS-E}$	14	13	12	13	13	13	16	18	13
$\underline{M}_{BPS-VE}$	19	18	18	20	22	18	26	29	18
$\underline{M}_{BPS-S}$	12	12	12	12	13	11	17	19	11
	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R
$\underline{M}_{BPS-E}$	22	32	29	35	70	36	47	113	41
$\underline{M}_{BPS-VE}$	26	38	27	39	86	45	61	150	58
$\underline{M}_{BPS-S}$	19	29	21	28	55	28	42	81	31

TABLE II.2.9: Number of iterations using preconditioner based on sparse Schur built using inexact local solvers $ILL^T(t)$.

In Table II.2.10, we display the iteration numbers required for the solution of the model Problem (II.2.16). For the $\varepsilon = 1$ case, the results are similar to those of other authors [45].

$\varepsilon = 1.0$				
	# domains			
Preconditioner	4×4	8×8	16×16	32×32
$\tilde{M}^{vh(1)}_{BPS-E}$	10	11	11	10
$\tilde{M}^{vh(2)}_{BPS-E}$	10	10	10	10
$\tilde{M}^{rb(1)}_{BPS-E}$	10	11	11	10
$\tilde{M}^{rb(2)}_{BPS-E}$	10	10	10	10
$\varepsilon = 10^{-4}$				
	# domains			
Preconditioner	4×4	8×8	16×16	32×32
$\tilde{M}^{vh(1)}_{BPS-E}$	105	394	606	765
$\tilde{M}^{vh(2)}_{BPS-E}$	105	394	606	765
$\tilde{M}^{rb(1)}_{BPS-E}$	28	49	81	96
$\tilde{M}^{rb(2)}_{BPS-E}$	28	49	81	96

TABLE II.2.10: Number of iterations varying the number of domains which size 16×16.

Namely, the number of iterations of the preconditioned conjugate gradient is independent of the number of domains as predicted by the theoretical estimation of the condition number given by Equation (II.2.1). Further, all the different approaches give nearly identical behavior and the bandwidth of the probing matrices does not greatly affect the convergence behavior.

For $\varepsilon = 10^{-4}$, however, these observations are no longer true. In particular, the number of iterations required to solve the anisotropic problem is significantly greater than that required for the Poisson problem. Further, the number of iterations increases for all the preconditioners as the number of subdomains is increased. While none of these methods is 'optimal', the BPS-red/black probing method clearly outperforms the other in terms of iterations.

To see the convergence degradation as a function of ε, we fix the number of domains at 8×8 and vary ε. The results given in Table II.2.11 illustrate the degradation as a function of ε. From

	ε					
Preconditioner	1.0	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
$\tilde{M}^{vh(1)}_{BPS-E}$	12	30	80	171	390	1069
$\tilde{M}^{vh(2)}_{BPS-E}$	12	25	65	161	388	1066
$\tilde{M}^{rb(1)}_{BPS-E}$	12	17	25	40	55	55
$\tilde{M}^{rb(2)}_{BPS-E}$	11	16	25	40	55	55

TABLE II.2.11: Number of iterations varying the anisotropic behavior on a 257×257 grid with a 8×8 decomposition.

an iteration point of view, it is apparent that the red/black probing is significantly better than the other technique for $\varepsilon \leq 10^{-3}$.

The alternating line probing scheme was implemented and tested first on Problem (II.2.16). In Table II.2.12 we illustrate the convergence for a box decomposition. For the $\varepsilon = 1$ case, there is

$\varepsilon = 1.0$				
	# domains			
Preconditioner	4×4	8×8	16×16	32×32
$\tilde{M}^{rb(1)}_{BPS-E}$	10	11	11	10
$\tilde{M}^{(1)}_{BPS-AL}$	10	10	10	10
$\varepsilon = 10^{-4}$				
	# domains			
Preconditioner	4×4	8×8	16×16	32×32
$\tilde{M}^{rb(1)}_{BPS-E}$	28	49	81	96
$\tilde{M}^{(1)}_{BPS-AL}$	11	13	16	20

TABLE II.2.12: Number of iterations varying the number of domains which size 16×16.

virtually no advantage to using the alternating line probing technique. However, when $\varepsilon = 10^{-4}$, this technique converges much faster than the other scheme (4 times faster for 1024 domains) with a convergence rate that is relatively independent of the number of domains. Similar results are given for a fixed domain size and varying ε in Table II.2.13. More specifically, the method is relatively insensitive to variations in ε though there is a peak which occurs between 10^{-2} and 10^{-3}. This peak can be predicted by applying Fourier analysis techniques to a closely related model problem. Essentially, alternating line probing is least effective when horizontal and vertical coupling among horizontal interfaces in the Schur complement stencil is approximately the same. This gives rise to a convergence peak as the anisotropic strength is varied (see [86] for details).

We conclude with two nonconstant coefficient examples where the direction of the anisotropic behavior changes within the domain. Specifically, we consider the following problem

$$\varepsilon_x(x,y)a(x,y)u_{xx} + \varepsilon_y(x,y)b(x,y)u_{yy} = f$$

	ε					
Preconditioner	1.0	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
$\tilde{M}_{BPS}^{rb(1)}$	12	17	25	40	55	55
$\tilde{M}_{BPS-AL}^{(1)}$	10	14	23	23	15	12

TABLE II.2.13: Number of iterations varying the anisotropic behavior on a 257×257 grid with a 8×8 decomposition.

defined on the unit square with Dirichlet boundary conditions. The functions $\varepsilon_x()$ and $\varepsilon_y()$ are depicted in Figure II.2.17 and II.2.18 for two different examples.

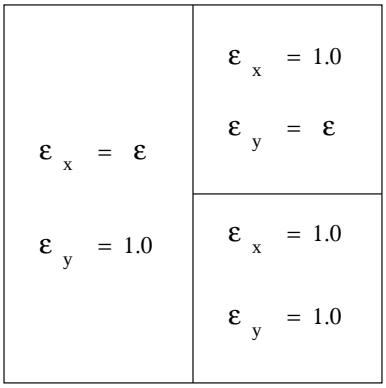


FIGURE II.2.17: Example (1).

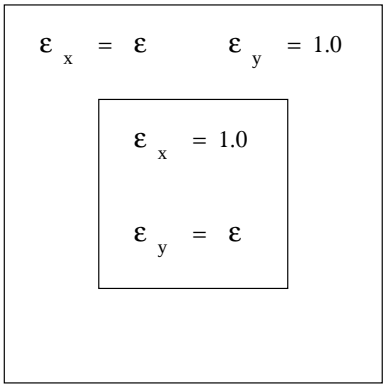


FIGURE II.2.18: Example (2).

Example (1)				
	# domains			
Coefficients	4×4	8×8	16×16	32×32
$a() = 1 + 4\sin^2(\pi(x+y))$ $b() = 1 + \cos^2(\pi(x+y))$	14	22	22	25
$a() = 1$ $b() = 1$	14	18	20	20
Example (2)				
	# domains			
Coefficients	4×4	8×8	16×16	32×32
$a() = 1 + 4\sin^2(\pi(x+y))$ $b() = 1 + \cos^2(\pi(x+y))$	22	35	49	77
$a() = 1$ $b() = 1$	22	31	40	53

TABLE II.2.14: Number of iterations of $\tilde{M}_{BPS-AL}^{(1)}$ on Example (1) and (2) varying the number of subdomains which size 16×16 - $\varepsilon = 1.0^{-4}$.

The results obtained for $\varepsilon = 10^{-4}$ and two different choices of $a()$ and $b()$ are given in Table II.2.14. Similar to the model problem, it is clear that the alternating line probing method is

superior in iterations. On Example (1) the general behavior of $\tilde{M}_{BPS-AL}$ is similar to that of the Poisson equation on a uniform grid in that the number of iterations does not depend strongly on the number of domains.

2.4.2 Two level preconditioners

Model problems

For the numerical experiments with the various coarse components, we consider model problems that have both discontinuous and anisotropic phenomena. Figure II.2.19 represents a unit square divided into six regions with piecewise constant functions g_j, $j = 1$ or 3. We consider the problems as having low intensity if $j = 1$ and high intensity if $j = 3$. Let a and b be the functions of the elliptic problem as described in Equation (II.2.3). Using this notation, we can define different problems with different degrees of difficulty. In the following description, the acronyms in capital letters are used to refer to the problems in Tables II.2.16 and II.2.17.

- high discontinuity **(HD)**: $a() = b() = g_3$,
- low discontinuity **(LD)**: $a() = b() = g_1$,
- high anisotropy **(HA)**: $a() = 10^3$ and $b() = 1$,
- low anisotropy **(LA)**: $a() = 10$ and $b() = 1$,
- high discontinuity and high anisotropy **(HDA)**: $a() = g_3$ and $b() = 1$,
- low discontinuity and low anisotropy **(LDA)**: $a() = g_1$ and $b() = 1$.

$g_j = 10^j$	$g_j = 1$	$g_j = 10^j$
$g_j = 1$	$g_j = 10^j$	$g_j = 1$

FIGURE II.2.19: Definition of two discontinuous functions on Ω, the unit square of $\mathbb{R}^2$.

For sake of comparison with classical BPS, we consider $M_{local} = M_E$ for all the experiments reported in the study of the proposed coarse components. The resulting two-level preconditioners are denoted in the tables:

sd: subdomain defined in Section 2.3.2,

vl: vertex-linear defined in Section 2.3.1 with the linear interpolation,

vo: vertex-operator-dependent defined in Section 2.3.1 with the operator-dependent interpolation,

vf: vertex-flat defined in Section 2.3.1 with the flat interpolation,

ed: edge defined in Section 2.3.3,

no: without coarse space in this case we only use the local preconditioner (block diagonal).

Experimental results

We study the numerical scalability of the preconditioners, by investigating the dependence of the convergence on the number and on the size of the subdomains. We remind our goal that is to obtain performance similar to those of the original BPS preconditioner even in presence of anisotropy or heterogeneity, with a simple algebraic structure. Firstly, in Table II.2.15, we give the figures for the five coarse spaces when solving the Poisson's problem using box decomposition. As we could expect, the behavior of all coarse-space options does not depend on the number of subdomains.

# subdomains	16	64	256	1024
no	15	28	48	90
sd	15	19	19	18
vf	15	18	18	18
vl	10	10	10	10
vo	10	10	10	10
ed	15	18	18	18

TABLE II.2.15: Number of PCG iterations varying the number of subdomains with 16×16 points per subdomain for Poisson's problem.

Dependency of the coarse preconditioners on H In Table II.2.16, we report the number of preconditioned conjugate gradient iterations for each model problem. For these tests, we vary the number of subdomains while keeping constant their sizes (i.e. H variable with $\frac{H}{h}$ constant). In this table each subdomain is a 16×16 grid and the number of subdomains goes from 16 up to 1024 using a box decomposition; that is 4×4 decomposition up to 32×32 decomposition.

	LA				LD				LDA			
# subdom.	16	64	256	1024	16	64	256	1024	16	64	256	1024
no	17	33	59	114	25	47	83	158	29	55	104	194
sd	18	25	27	28	19	19	19	19	22	30	33	34
vf	19	24	29	31	20	21	21	21	23	28	31	32
vl	15	17	17	17	13	13	12	12	14	16	17	17
vo	15	17	17	17	11	11	11	11	14	16	17	17
ed	19	26	27	28	20	20	18	18	21	26	27	28
	HA				HD				HDA			
# subdom.	16	64	256	1024	16	64	256	1024	16	64	256	1024
no	19	42	69	127	25	50	87	172	37	149	302	629
sd	30	64	75	86	18	19	19	19	30	64	81	83
vf	27	52	72	85	21	22	22	21	31	76	86	99
vl	26	45	66	73	16	18	18	16	21	63	81	89
vo	26	45	66	73	11	11	11	11	20	60	81	88
ed	24	43	57	69	17	19	19	18	31	62	70	77

TABLE II.2.16: Number of PCG iterations varying the number of subdomains with 16×16 points per subdomain.

In the first row, we see the growth of the number of iterations of a block Jacobi preconditioner

without using any coarse grid correction. Its numerical behavior is well known and is governed by the theoretical bound for its condition number (II.2.17).

For HD and LD the behavior of all coarse alternatives are similar to the one observed for Poisson's problem in Table II.2.15, that is, the number of iterations is independent of the number of subdomains. For LA and LDA, this similarity is still true for vertex-linear and vertex-operator-dependent, the three others exhibit a slight insignificant increase in the number of iterations.

For HA, the coarse spaces subdomain-based and vertex-flat do not improve the convergence for a number of subdomains less than 1024. For vertex-linear and vertex-operator the improvement is modest for 256 subdomains. It appears that on this example the condition numbers of the preconditioned linear systems with and without the coarse components are comparable, for instance, 2×10^2 with and 6×10^2 without the vertex-linear coarse component using 256 subdomains. More precisely, the smallest eigenvalue is not that affected by the use of the coarse component as it is for the other model problems. In the presence of high anisotropy (HA and HDA), the convergence rate of all the alternatives is comparable and depends on the number of subdomains, while an asymptotic behavior tends to appear when the number of subdomains increases.

Dependency of the coarse preconditioners on $\frac{H}{h}$ To study the sensitivity of the preconditioners to the size of the subdomains (i.e. $\frac{H}{h}$ variable with H constant), we report in Table II.2.17, the experiments observed with 256 subdomains (16×16 box decomposition) when the size of the subdomains varies from 8×8 up to 32×32.

	LA				LD				LDA			
size	64	256	1024	4096	64	256	1024	4096	64	256	1024	4096
no	66	69	78	96	73	83	106	133	97	104	119	150
sd	30	27	36	42	16	19	23	30	29	33	36	45
vf	25	29	32	35	18	21	24	31	27	31	34	40
vl	15	17	19	21	12	14	15	19	15	18	19	22
vo	15	17	19	21	9	11	12	16	15	17	19	22
ed	24	27	32	34	16	18	22	29	24	27	31	36
	HA				HD				HDA			
size	64	256	1024	4096	64	256	1024	4096	64	256	1024	4096
no	66	69	73	76	78	87	116	141	280	302	297	389
sd	65	75	76	76	17	19	23	31	65	81	87	96
vf	66	72	75	76	20	22	25	31	80	86	89	91
vl	60	64	64	63	16	18	18	23	79	81	80	77
vo	60	66	64	63	10	11	13	16	77	81	79	80
ed	51	57	59	63	16	19	22	29	63	70	79	86

TABLE II.2.17: Number of PCG iterations varying the grid size of the subdomains from (8×8) up-to (64×64) using a (16×16) decomposition.

We observe that the convergence of all the preconditioners depend slightly on the size of the subdomains. Furthermore, in the anisotropic experiments (HA and HDA), this dependence is surprisingly negligible for the vertex-linear and for the vertex-operator-dependent alternatives. The number of iterations of those two preconditioners tends to be stable around 64 and 80 for the problems HA and HDA, respectively.

On problems that are not highly anisotropic, all the coarse-components give rise to preconditioners that are independent of on weakly dependent on the number of subdomains and that have

a mildly dependence on the size of the subdomains.

If we compare the LD and HD columns, we see that all the preconditioners but vertex-linear are quite insensitive to discontinuity. Further, the vertex-operator-dependent alternative, specifically designed to handle interfaces that cross a discontinuity, has the best performance.

The numerical experiments tend to show that the most dominating component is not the granularity of the coarse-space (the finest is not necessarily the best) but the restriction/interpolation operator R_0 as show the experiments with vertex. The restriction operator governs, in most cases, the quality of the coarse representation of the complete equation. The flat interpolation operator is always the worst and operator-dependent behaves the best on problems with discontinuities for whose it was designed. For discontinuous problems, LD and HD, vertex-operator-dependent is the most efficient while for highly anisotropic problems, HA and HDA, edge is the most efficient for all cases but one. For LA and LDA operator-dependent and vertex-linear had the same behavior. The good performance of edge on anisotropic problems can be explained by the fact that we consider anisotropies aligned with the discretization and because we use regular box decompositions, two opposite edges E_i of a subdomain are strongly coupled. The edge coarse space captures the strong coupling while the other alternatives mix, therefore miss, this information. This latter behavior is related to the fact that the supports of the basis vectors of all coarse spaces, but edge, contain at least two weakly coupled edges. So, the transfer operators are not able, in this specific case, to retrieve and to spread the most appropriate information.

2.5 Concluding remarks

We have introduced three new local preconditioners and a set of coarse space components to build two-level preconditioners. The local preconditioners are based on an explicit computation of the local Schur complement matrices but computing alternatives have been proposed to overcome this possible drawback.

The first local preconditioner, M_{VE}, aims at recovering some information relative to the interface nodes close to the vertices of the coarse mesh τ^H defined by the decomposition. This preconditioner shows some advantages over the simple block Jacobi preconditioner M_E for the solution of linear systems arising in the solution of parabolic problems where one-level preconditioners might be scalable [38, 126]. For the solution of such linear systems, M_{VE} is a cheap alternative to improve the simple block Jacobi preconditioner. Both have similar computational complexity and parallel performance [35]. In addition, the use of approximate local Schur complement does not penalize significantly the numerical behavior of M_{VE}. These advantages vanish for the solution of elliptic problems when, to ensure the numerical scalability, the coarse space preconditioner component smoothes its effect compared to M_E. For those elliptic problems, the use of approximate local solvers affect significantly the numeriacl behavior of the resulting preconditioner $\tilde{M}_{BPS-VE}$.

The second local preconditioner, closely related to the Neumann-Neumann preconditioner, demonstrates a very attractive numerical behavior on heterogeneous and anisotropic equations. These problems appear, for instance, in the solution of the drift-diffusion equations involved in semi-conductor device modeling. An efficient implementation of the local Schur complement construction directly benefit from the ongoing development of advanced sparse direct solvers like MUMPS [4]. One of the new capability of this sparse direct solver is to compute the Schur complement of a given list of variables in a given matrix; this coincide exactly with our need in non-overlaping domain decomposition. However, we propose an alternative based on approximated local Schur complements built thanks to incomplete Cholesky factorizations. Based on an extensive benchmarking, we show that the resulting preconditioner, with a cheap construction, retains the main numerical features of M_{BPS-S}.

We have adapted the domain decomposition probing method for highly anisotropic equations.

Specifically, we have considered two variants of the probing BPS type preconditioner in [45]. One variant, BPS-red/black probing, uses a multicolor ordering of the edges to obtain suitable probe vectors to construct good approximations to the diagonal blocks of the Schur complement. The primary idea is to avoid mixing information between neighboring interfaces, which are often strongly coupled for anisotropic problems when forming probe approximations. Though described for structured meshes, the same coloring technique can be easily extended to unstructured meshes. We have shown through experimental results that the BPS-red/black probing preconditioner significantly outperforms the BPS-probing, BPS-Fourier [45], and balanced Neumann-Neumann preconditioners on anisotropic problems [89]. Despite these improvements, the convergence associated with all these methods including the BPS-red/black probing scheme still deteriorates when either the anisotropy or the number of subdomains is increased. To overcome this drawback, we have proposed a variant for structured meshes, $\tilde{M}_{BPS-AL}$, that introduces a series of band matrices. Each band matrix corresponds to the restriction of the Schur complement to a line of the original discretization grid. The $\tilde{M}_{AL}$ probing looks somewhat-like an alternating line relaxation procedure and was designed to approximate the coupling between neighboring interfaces properly. Experimental results show that with this new preconditioner,$\tilde{M}_{BPS-AL}$, the number of iterations only weakly depends on the number of subdomains (when the number of points per subdomain is fixed) as well as on the anisotropy. Though this preconditioner costs a bit more to construct and apply, it often requires significantly less iterations than the BPS-red/black probing method.

Finally we have presented two-level preconditioners for Schur complement domain decomposition methods in two dimensions built by combining a variant of the of the local component of the BPS preconditioner with a set of new algebraic coarse space components. They are defined using the mesh partitioning information and simple interpolation operators that follow a partition of unity principle. We have illustrated their numerical behaviour on a set of two-dimensional model problems that have both anisotropy and discontinuity. Those experiments tend to show that the most dominating component is not the the granularity of the coarse-space (the finest is not necessarily the best) but the restriction/interpolation operator R_0. This operator governs, in most cases, the quality of the coarse representation of the complete equation. The experiments have been performed on regular meshes but there is no limitation for the implementation of the proposed two-level preconditioners on unstructured grids, whereas the possible rank deficiency, that appears in the domain-coarse alternative, could be more tricky to discard.

Chapter 3

Some investigations of overlapping domain decomposition method in computational fluid dynamics

3.1 Introduction

In the past years domain decomposition for linear partial differential equations have graduated from theory into practice in many applications as shown by the evolution of the papers in the proceedings of the series of conferences dedicated to that topic [21, 41, 42, 90, 91, 92, 111, 112, 115, 121, 135]. In particular the Newton-Krylov and Krylov-Schwarz schemes have begun to become established techniques in computational fluid dynamics over the past decade. On the one hand Newton-Krylov methods [27, 28] are potentially well suited and increasingly popular for the implicit solution of nonlinear problems whenever the computation of the true Jacobian is too expensive; on the other hand the Krylov-Schwarz methods are robust domain decomposition algorithms for solving linear partial differential equations [60, 61]. Combining the two above algorithms leads to a family of algorithms called Newton-Krylov-Schwarz methods that are general purpose parallel solvers for nonlinear partial differential equations. This technique and closely related variants have been applied to computational fluid dynamics applications [29, 30, 31, 132].

In this chapter we present some investigations on additive Schwarz preconditioners for a Krylov-Schwarz domain decomposition algorithm for the finite element solution of the nonlinear Navier-Stokes equations. The nonlinear part is tackled with a variant of the Newton-Krylov method. We omit the description of the fluid dynamics equations, discretization and numerical schemes, that are out of the scope of this study, and refer the reader to [106, 107, 157] for a complete description of these computational fluid dynamics aspects. We rather focus on the description of the preconditioners to solve the linear systems involved at each step of the nonlinear iterations.

This study was conducted in the framework of an industrial collaboration with Dassault-Aviation that aimed at introducing efficient preconditioners in an existing parallel code that computes the steady state solution of the Navier-Stokes equations on large 3D unstructured finite element meshes. In the original code the linear systems are first preconditioned by a symmetric block diagonal scaling which purpose is to a-dimension the variables associated with each vertex of the discretization (5 degrees of freedom per vertex); the preconditioned system is explicitly formed and a restarted GMRES [142] is applied for its solution. The parallelism implemented in the code exploited an element-based partition of the finite element mesh. In this context, domain

decomposition algorithms are natural candidates to precondition the linear systems and Schwarz methods are particularly well suited. Indeed the constraints imposed by the size of the local subdomain make them too large to afford direct methods for solving the local problems. Therefore this prevents us to envisage any Schur complement approach. However Schwarz techniques with inexact local solves can be considered.

This chapter is organized as follows. In the next section we briefly present the classical Schwarz methods. In Section 3.3, we describe the variant of the additive Schwarz we have considered and report numerical experiments in Section 3.4.

3.2 The Schwarz procedure

The earliest known domain decomposition method was introduced by H. A. Schwarz in 1890 [145] to prove the existence of harmonic functions on irregular regions which are the union of overlapping subregions. The interest in this method was renewed because many of its variants enable to express a coarse grain parallel algorithms suitable for modern distributed memory computers. To describe the classical alternating Schwarz procedure let us consider the classical 2D Poisson equation:

$$\left\{ \begin{array}{rcll} -\Delta u & = & f & \text{in } \Omega, \\ u & = & 0 & \text{on } \partial\Omega, \end{array} \right. \tag{II.3.1}$$

and note

$$Ax = b, \tag{II.3.2}$$

the linear system resulting from the discretization of (II.3.1) by either finite differences or finite elements.

We consider two overlapping subregions $\{\Omega_1, \Omega_2\}$ such that $\Omega_1 \cup \Omega_2 = \Omega$, $\Gamma_1 = \partial\Omega_1 \cap \Omega_2$, $\Gamma_2 = \partial\Omega_2 \cap \Omega_1$, and denote $u_i = u|_{\Omega_i}$. Given an initial guess u^0 the iterate u^{n+1} is determined from u^n by sequentially updating the approximated solution in the two subregions:

$$\left\{ \begin{array}{rcll} -\Delta u_1^{n+1} & = & f & \text{in } \Omega_1, \\ u_1^{n+1} & = & u_2^n|_{\Gamma_1} & \text{on } \Gamma_1, \\ u_1^{n+1} & = & 0 & \text{on } \partial\Omega_1 \backslash \Gamma_1, \end{array} \right.$$

and

$$\left\{ \begin{array}{rcll} -\Delta u_2^{n+1} & = & f & \text{in } \Omega_2, \\ u_2^{n+1} & = & u_1^{n+1}|_{\Gamma_2} & \text{on } \Gamma_1, \\ u_2^{n+1} & = & 0 & \text{on } \partial\Omega_2 \backslash \Gamma_2. \end{array} \right.$$

The iterate u^{n+1} is then defined by:

$$u^{n+1}(x,y) = \left\{ \begin{array}{l} u_2^{n+1}(x,y) \text{ if } (x,y) \in \Omega_2, \\ u_1^{n+1}(x,y) \text{ if } (x,y) \in \Omega \backslash \Omega_2. \end{array} \right.$$

It is straightforward to extend the above alternating Schwarz procedure to discretization of (II.3.1). To simplify the exposure, we will mix in the sequel continuous surfaces and curves with sets of indices. This ambiguity can be disregarded if we consider that, in order to minimize notation, the symbols Ω_i and $\partial\Omega_i$ represent either continuous sets or discrete sets of indices associated with the grid points. We shall describe the discrete algorithm in matrix notation and denotes R_i^T the extension operator which extends by zero in $\Omega \backslash \Omega_i$ a vector of nodal values in Ω_i. The transpose R_i of this extension map is a canonical restriction whose action restricts a full vector of nodal values in Ω to a vector whose entries are those associated with the nodes in Ω_i. The

discretization of (II.3.1) on Ω_i with Dirichlet boundary conditions on $\partial\Omega_i \backslash \partial\Omega$ defines the local Dirichlet matrices:

$$A_i = R_i A R_i^T.$$

The discrete version of the alternating Schwarz method to solve (II.3.2) starts from an initial guess x^0 and generates the sequence of iterates:

$$\begin{aligned} x^{n+1/2} &= x^n + R_1^T A_1^{-1} R_1 (b - Ax^n), \\ x^{n+1} &= x^{n+1/2} + R_2^T A_2^{-1} R_2 (b - Ax^{n+1/2}). \end{aligned}$$

This can be interpreted as a generalization of a block Gauss-Seidel iteration for solving (II.3.2) with overlapping blocks. Defining

$$P_i \equiv R_i^T A_i^{-1} R_i \ ,$$

it can be shown [117] that the iteration matrix is $(I - P_2)(I - P_1)$ hence this scheme is usually called a *multiplicative* Schwarz iteration.

An analogous block Jacobi scheme can be defined by

$$\begin{aligned} x^{n+1/2} &= x^n + R_1^T A_1^{-1} R_1 (b - Ax^n), \\ x^{n+1} &= x^{n+1/2} + R_2^T A_2^{-1} R_2 (b - Ax^n). \end{aligned}$$

Notice that eliminating $x^{n+1/2}$ gives rise to

$$x^{n+1} = x^n + (R_1^T A_1^{-1} R_1 + R_2^T A_2^{-1} R_2)(b - Ax^n)$$

that is a preconditioned Richardson method, where the preconditioner $P_1 + P_2$ is referred to as *additive* Schwarz preconditioner.

It can be remarked that for many subdomains and similarly to the block Gauss-Seidel and block Jacobi methods, the additive Schwarz iteration is naturally parallelizable while coloring techniques should be used to parallelize the multiplicative Schwarz method. In addition both techniques are usually not implemented as stationary methods but are rather accelerated by Krylov iterations. Unfortunately for elliptic problems the two resulting preconditioners are not scalable in the sense that the condition number of the precondition matrices increases when the number of subdomains is increased. Similarly to preconditioners for the Schur complement, the numerical scalability can be obtained thanks to the use of additional coarse space components. We refer to [44, 61, 147] and the references therein for a more detailed presentation of the Schwarz methods.

3.3 Some variants of additive Schwarz preconditioner

Recently a non-symmetric variant of the additive Schwarz preconditioner has been introduced in [32] and applied to 3D flow simulations in [29]. It is referred to as restricted additive Schwarz preconditioner and can be described as follows. Let $\mathcal{M}$ be an unstructured finite element mesh. Using an element-based partitioning, the mesh $\mathcal{M}$ can be decomposed into N disjoint sets of elements, that define the subdomains, or equivalently into N overlapping sets of vertices. The elements on the border between subdomains share some vertices with their neighbors, these nodes are called "interface vertices". Let denote $\{\Omega_i\}_{i=1,N}$ the overlapping subsets of vertices and Ω the complete set of vertices over the entire mesh. We have $\Omega = \bigcup_{i=1}^N \Omega_i$, which can be called a "partition with a minimum overlap" of Ω with respect to the fact that $\{\Omega_i\}$ is a partition of the elements of Ω. From the element partitioning, we can obtain a vertex partitioning by allocating each interface vertex to a single subset of elements. The resulting partitioning of the vertices is denoted $\Omega_i^0 \subset \Omega_i$, with $\Omega = \bigcup_{i=1}^N \Omega_i^0$ and $\Omega_i^0 \cap \Omega_j^0 = \emptyset$ for $i \neq j$. Let R_i^0 define the canonical

restriction whose action restricts a full vector of nodal values in Ω to a vector whose entries are those associated with the nodes in Ω_i^0. With these notations we can define the classical additive Schwarz preconditioner

$$M_{AS} = \sum_{i=1}^{N} R_i^T A_i^{-1} R_i \ , \tag{II.3.3}$$

and the restricted additive Schwarz preconditioner

$$M_{RAS} = \sum_{i=1}^{N} (R_i^0)^T A_i^{-1} R_i \ . \tag{II.3.4}$$

Because the exact solution of the local problems that appear in M_{AS} (II.3.3) and M_{RAS} (II.3.4) cannot be afforded in time and memory for 3D industrial problems, we replace them by approximate solution obtained from block $ILU(0)$. In addition, to reduce the redundant computation and the amount of data to exchange, we only consider one element overlap between the subdomains. In this respect the local Dirichlet matrices are simply built by assembling the local stiffness matrices at the vertices on the interface between the subdomains. On unstructured meshes with one element overlap, the local stiffness matrix built from the local subdomain elements have often the same pattern as the local Dirichlet matrix. This is particularly convenient as the data structure to represent the connectivity of both matrices are the same. Unfortunately when the interface is not smooth the two patterns differ. This is illustrated in 2D by Figure II.3.1, where the connectivity between vertex 1 and 2 does not exist in the stiffness matrix computed on Ω_1 but is present in the local Dirichlet matrix resulting from an one element overlap between Ω_1 and Ω_2. In that case building the local Dirichlet matrix becomes more complex as the connectivity on the overlap region should be built first and more memory consuming as a complete separate data structure should be set-up to represent the Dirichlet problem A_i. To overcome this difficulty, we consider variants of M_{AS} and M_{RAS}, where we only assemble the diagonal blocks associated with the vertices on the interface and not the off-diagonal blocks that represent the coupling between the interface vertices.

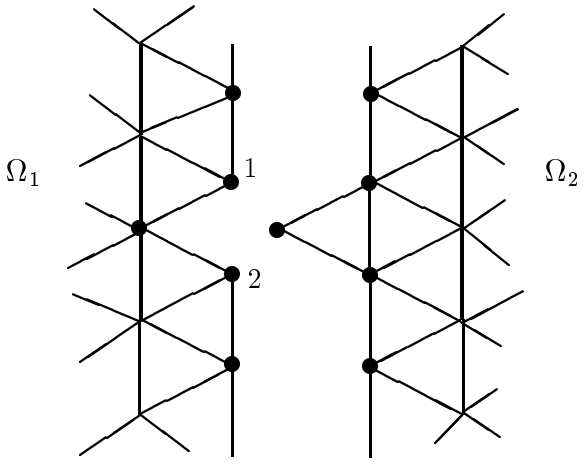


FIGURE II.3.1: A 2D example of non-smooth interface.

In the next section we report experiments with three preconditioners:

- $M_{ILU(0)-AS}$: Additive Schwarz with an inexact local solution using ILU(0),
- $M_{ILU(0)-dAS}$: Additive Schwarz with inexact local solution using ILU(0) and only assembling of the diagonal block of the interface vertices,
- $M_{ILU(0)-dRAS}$: Restricted Additive Schwarz with inexact local solution using ILU(0) and only assembling the diagonal block of the interface vertices.

These preconditioners have been implemented to precondition the restarted GMRES Krylov solver that is applied to the linear systems involved in the nonlinear steps.

3.4 Numerical experiments

In this section we only report some numerical experiments with the ONERA M6 wing to test 3D turbulent flows. The mesh is composed by 77000 vertices and decomposed into 4, 8, 16 or 32 subdomains. The size of the global problem prevents us to perform the sequential simulation to use it as reference. The restart for GMRES was equal to 20. The tolerance for the stopping criterion is defined by the ratio of the 2-norm of the residual divided by the 2-norm of the right hand side and is set to 10^{-1}. All the experiments were performed in double precision arithmetic.

The time constraints to perform this industrial collaboration did not enable us to perform exhaustive experiments but rather incremental comparisons between the various Schwarz variants. In this section we do not report any result with $M_{ILU(0)-AS}$ because the code we have implemented was not robust enough to handle non-smooth interfaces. However for smooth interfaces we observed that $M_{ILU(0)-AS}$ was slightly more efficient than $M_{ILU(0)-dAS}$. The gap was not big enough to justify neither the significant manpower effort to develop a code able to handle completely the overlap connectivity, nor to afford for the extra memory required to store the local Dirichlet matrices in fully separate data structures.

In Figure II.3.2 we depict the 2-norm of the nonlinear residual as a function of the nonlinear iteration for a run with 4 subdomains without preconditioner and with $M_{ILU(0)-dAS}$. As it can be seen a significant gain is introduced by $M_{ILU(0)-dAS}$. The main improvement is due to the reduction of the number of nonlinear iterations required to compute the steady state solution. This is due to the fact that it permits to use a larger CFL, enabling to quickly compute the steady state solution, as for both simulations we use the maximum acceptable CFL.

Figure II.3.3 illustrates the reasonable scalability of the preconditioner. Although the total number of Krylov iterations increases when the number of subdomains is increased, the increase is still moderate when moving from 4 to 16 subdomains. On a pure elliptic problem the number of iterations with 16 subdomains would be twice larger than with 4 subdomains.

In Figure II.3.4 we compare the behavior $M_{ILU(0)-dAS}$ and $M_{ILU(0)-dRAS}$ for a 4 subdomain decomposition. The same trend is observed on other decompositions, that is, $M_{ILU(0)-dRAS}$ exhibits a better convergence rate than $M_{ILU(0)-dAS}$ for the solution of the linear systems. We mention that both preconditioners enable to use the same CFL condition and follow the same nonlinear convergence path. It should also be mentioned that the decision algorithm to build Ω_i^0 from Ω_i by logically assigning an vertex interface to a subdomain does not use any numerical information. We arbitrary allocates an interface vertex to the subdomain that has the larger number in the ordering of the subdomains.

Finally in Figure II.3.5 we show the numerical scalability of $M_{ILU(0)-dRAS}$. Surprisingly on that example the number of iterations does not increase when we increase the number of subdomains from 4 to 16 but even decreases. This decrease is not longer observed when we go from 16 to 32 subdomains. Although this behavior is not fully representative from the set of experiments we have performed, we observed that $M_{ILU(0)-dRAS}$ exhibits a better numerical scalability than $M_{ILU(0)-dAS}$.

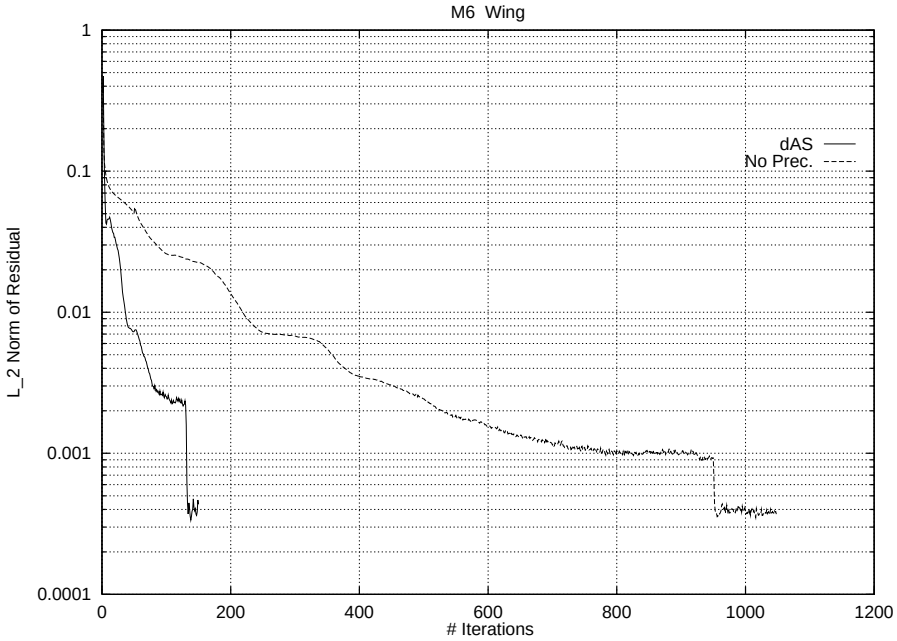


FIGURE II.3.2: 4 subdomains decomposition
dashed: Unprecond. , CFL=5 - solid: $M_{ILU(0)-dAS}$, CFL=50.

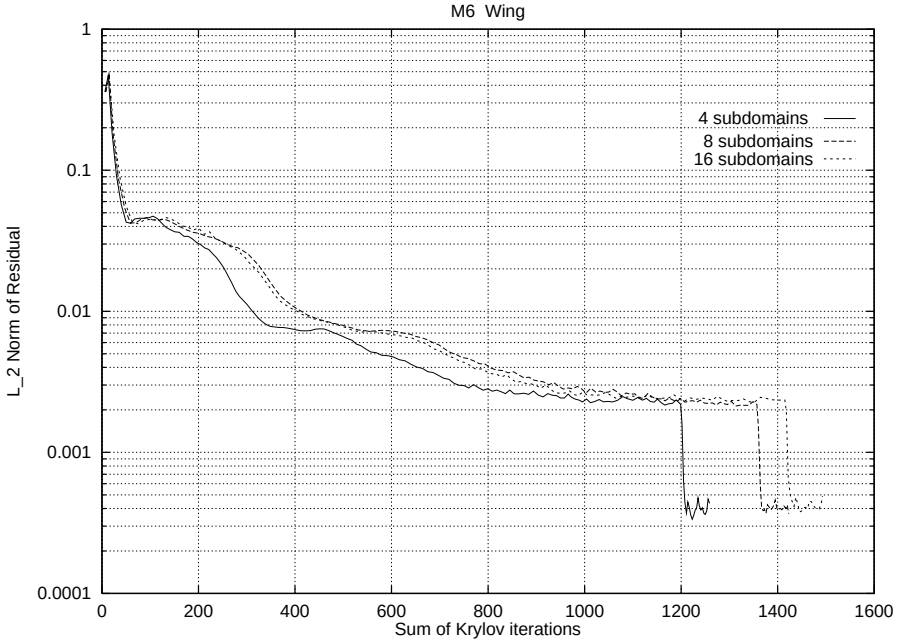


FIGURE II.3.3: 2-norm of the nonlinear residual v.s. sum of Krylov iterations with $M_{ILU(0)-dAS}$ varying the number of subdomains, CFL=50.

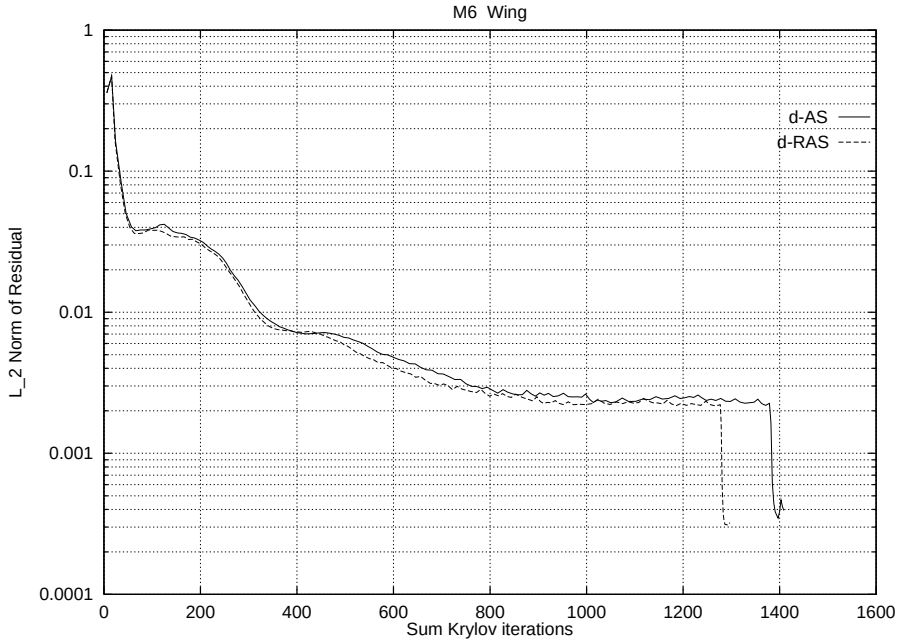


Figure II.3.4: 4 subdomains decomposition, CFL = 50
$M_{ILU(0)-dAS}$ (solid) - $M_{ILU(0)-dRAS}$ (dashed).

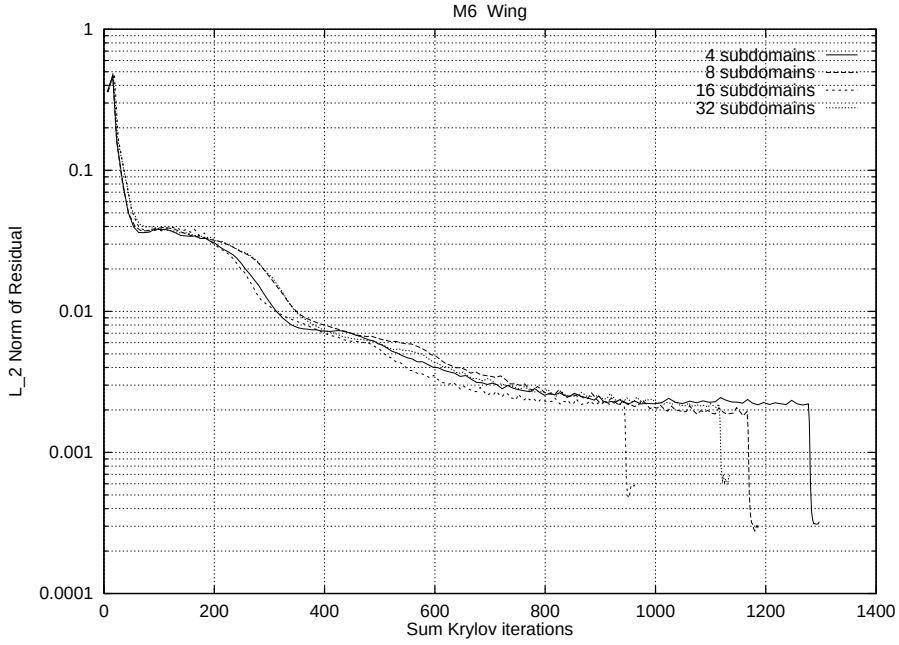


Figure II.3.5: 2-norm of the nonlinear residual v.s. sum of Krylov iterations with $M_{ILU(0)-dRAS}$ varying the number of subdomains, CFL=50.

3.5 Concluding remarks

It is important to notice that during this study performed in a relative short elapsed time it was possible to adapt a semi-industrial aerodynamic code in order to quickly investigate some domain decomposition ideas and preconditioners for practical simulation using coarse grained parallelizable techniques. We are aware that adding a coarse space component to the considered preconditioners should improve the convergence rate of the preconditioned GMRES as shown in [1, 30]. Unfortunately we did not have the time to investigate such a possibility in the framework of this collaboration.

Chapter 4

Sparse approximate inverse for boundary element methods in electromagnetism

4.1 Introduction

In recent years, there has been a significant amount of work on the simulation of electromagnetic wave propagation phenomena, addressing various topics ranging from radar cross section to electromagnetic compatibility, to absorbing materials, and antenna design. To address these problems the Maxwell equations are often solved in the frequency domain leading to singular integral equations of the first kind. The discretization by the boundary element method (BEM) results in linear systems with dense complex matrices which are very challenging to solve. With the advent of parallel processing, this approach has become viable for large problems and the typical problem size in the electromagnetics industry is continually increasing.

In this chapter, we consider the solution of linear systems of the form

$$Ax = b$$

where the coefficient matrix $A = [a_{ij}]$ is a large, dense, complex matrix of order n arising from the discretization. The coefficient matrix can be symmetric non-Hermitian in the EFIE (Electric Field Integral Equation) formulation, or unsymmetric in the CFIE (Combined Field Integral Equation) formulation. The unknowns in the vector x are associated with the edges of an underlying mesh on the surface of the object. In this chapter, we will only consider numerical examples where A is symmetric because EFIE usually gives rise to linear systems that are more difficult to solve with iterative methods. The techniques considered here can be applied equally well to unsymmetric matrices. In fact in the numerical experiments we use non-symmetric solvers because the preconditioners that we construct are unsymmetric. We can, of course, construct either only the lower or only the upper part of the preconditioner and use a symmetric preconditioner obtained by reflecting this in the diagonal. One problem is that the resulting preconditioner depends on the ordering of the matrix. In previous tests [34], we investigated the effect of symmetrizing the preconditioner by averaging the off-diagonal entries after its construction and using such a symmetrized preconditioner with symmetric QMR but found that this caused a marked deterioration in the quality of the preconditioner leading to far more iterations of the iterative method. We plan to further investigate symmetric strategies in future work but, in this present study, we will stick with unsymmetric techniques and preconditioners.

Direct dense methods based on Gaussian elimination are often the method of choice because they are reliable and predictable both in terms of accuracy and cost. However, for large-scale problems, they become impractical even on large parallel platforms because they require storage of n^2 double precision complex entries of the coefficient matrix and $\mathcal{O}(n^3)$ floating-point operations to compute the factorization, where n denotes the size of the linear system. Iterative Krylov subspace based solvers are a promising alternative provided we have fast matrix-vector multiplications and robust preconditioners. There are active research efforts on multipole techniques to perform fast matrix-vector products with $\mathcal{O}(n\log(n))$ computational complexity including strategies for parallel distributed memory implementations (see [55, 56, 57]). In this chapter, we focus on the other key component of Krylov methods in this context; that is, we study the design of robust preconditioning techniques.

The parallel framework suggests that sparse approximate inverses based on Frobenius-norm minimization techniques are promising candidates for the efficient preconditioning of these systems. Such techniques exhibit a good level of numerical efficiency on this class of applications when compared with the implicit approach based on incomplete factorization (see [34], [47]). The normal requirement for a good preconditioner is that it is easy to construct, cheap to store and to apply, is parallelizable and, of course, is effective in accelerating the convergence of Krylov solvers. To be computationally affordable on dense linear systems, Frobenius-norm minimization preconditioning techniques require a suitable strategy to identify the relevant entries to consider in the original matrix A, in order to define small least-squares problems, as well as an appropriate sparsity structure for the approximate inverse.

For sparse matrices, two strategies can be used to define the sparsity structure of the preconditioner. A dynamic approach constructs the nonzero pattern of the preconditioner by monitoring the residual in the least-squares problems during the computation. This is generally effective but is usually very expensive. A static approach that requires an *a priori* nonzero pattern for the preconditioner, introduces significant scope for parallelism and has the advantage that the memory storage requirements and computational cost for the setup phase are known in advance. However, it can be very problem dependent.

In this chapter, we propose some new efficient static nonzero pattern selection strategies both for the preconditioner and for the selection of the entries of A in order to develop robust preconditioners for applications in electromagnetism. In this chapter we consider a set of test examples, arising from both academic and industrial applications that are representative of the general numerical behavior that we observed. Those examples are similar to those considered in [34], where all test problems are of the same order or even smaller; larger problems will be examined when we will move to multipole method. More specifically, we here consider the following geometries where, for physical consistency, we have set the frequency of the wave so that there are about ten discretization points per wavelength [12]:

Example 1: a cylinder with a hollow inside, a matrix of order $n = 1080$, see Figure 4.1(a);

Example 2: a cylinder with a break on the surface, a matrix of order $n = 1299$, see Figure 4.1(b);

Example 3: a satellite, a matrix of order $n = 1701$, see Figure 4.1(c);

Example 4: a parallelepiped, a matrix of order $n = 2016$; and

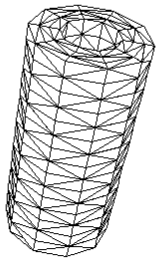
Example 5: a sphere, a matrix of order $n = 2430$.

We perform experiments with the following Krylov solvers:

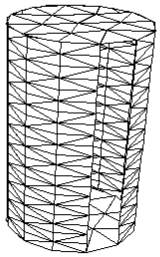
- restarted GMRES [142];
- Bi-CGSTAB [152];

- symmetric and unsymmetric QMR [75];
- TFQMR [73].

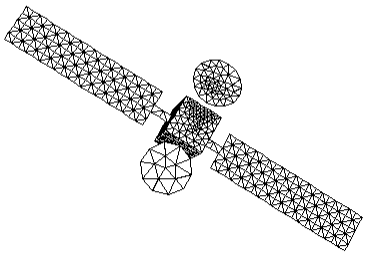
In Section 4.2, we describe the construction of the preconditioner using our proposed static pattern strategies and report on the associated numerical experiments. Finally, in Section 4.3, we present some remarks arising from the work.



(a) Example 1



(b) Example 2



(c) Example 3

Figure II.4.1: Mesh associated with test examples.

4.2 Static pattern selection and dropping strategies

Frobenius-norm minimization is one of the most natural approaches for building explicit preconditioners. The idea is to compute the sparse approximate inverse as the matrix M which minimizes $\|I - MA\|_F$ (or $\|I - AM\|_F$ for right preconditioning) subject to certain sparsity constraints. The Frobenius-norm is usually chosen since it allows the decoupling of the constrained minimization

problem into n independent linear least-squares problems, one for each column of M (when preconditioning from the right) or row of M (when preconditioning from the left). In our present applications, these least-squares problems are small enough to be solved using a dense QR decomposition. The independence of these least-squares problems follows immediately from the identity:

$$\|I - MA\|_F^2 = \|I - AM^T\|_F^2 = \sum_{j=1}^{n} \|e_j - Am_{j,*}\|_2^2 \tag{II.4.1}$$

where e_j is the jth unit vector and $m_{j,*}$ is the column vector representing the jth row of M. In the case of right preconditioning, the analogous relation

$$\|I - AM\|_F^2 = \sum_{j=1}^{n} \|e_j - Am_{*,j}\|_2^2 \tag{II.4.2}$$

holds, where $m_{*,j}$ is the column vector representing the jth column of M. Clearly, there is considerable scope for parallelism in this approach.

The main issue is the selection of the nonzero pattern of M. The idea is to keep M reasonably sparse while trying to capture the "large" entries of the inverse, which are expected to contribute the most to the quality of the preconditioner. For this purpose, two approaches can be followed: an adaptive technique that dynamically tries to identify the best structure for M; and a static technique, where the pattern of M is prescribed *a priori* based on some heuristics. Some early references to this latter class can be found in [13, 14, 15, 72] and in [2] for some applications to boundary element matrices in electromagnetism.

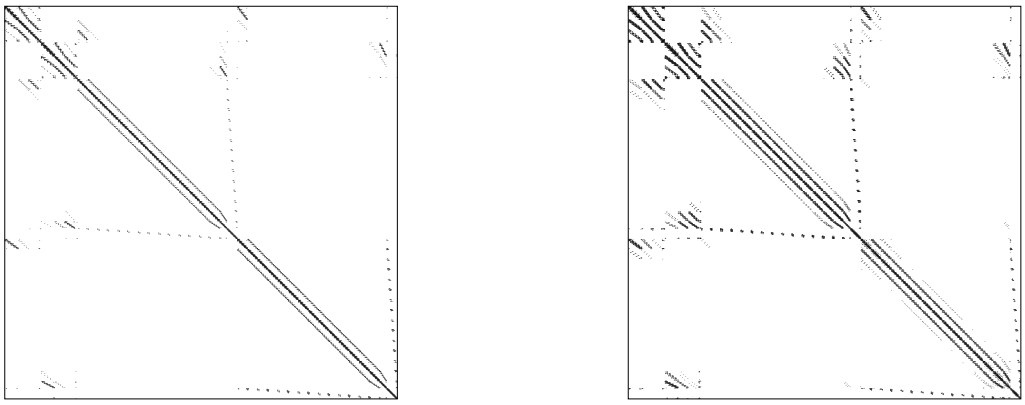
In addition, when the coefficient matrix is dense, the preconditioner should be constructed from a sparse approximation of A in order to reduce the computational cost of the least-squares solutions.

4.2.1 Strategies for the preconditioner

When the coefficient matrix has a special structure or special properties, for instance a banded matrix with a good degree of diagonal dominance or a banded SPD matrix, efforts have been devoted to find a pattern that can retain the entries of A^{-1} having large modulus, see [48] and [58] for example. Unfortunately, for general unstructured matrices, it is very difficult to predict a good pattern for the inverse in advance. Adaptive strategies that compute the pattern dynamically can provide very good preconditioners, even on hard problems, but at the cost of a very large amount of computing time and memory. In some cases it is possible to take advantage of special features of the underlying physical problem and compute a good *a priori* pattern for the approximate inverse.

Algebraic strategy

The boundary element method discretizes integral equations on the surface of the scattering object, generally introducing a very localized strong coupling among the edges in the underlying mesh. Each edge is strongly connected to only a few neighbors, while, although not null, far-away connections are much weaker. This means that a very sparse matrix can still retain the most relevant contributions from the singular integrals that give rise to dense matrices. Due to the decay of the Green's function, the inverse of A may exhibit a very similar structure to A as illustrated in Figure II.4.2 where we display the pattern of A and A^{-1} when the smallest entries are dropped. Thus, in this case, a good pattern for the sparse approximate inverse is likely to be the nonzero pattern of a sparse approximation of A, constructed by dropping all the entries lower than a prescribed threshold, as suggested for instance in [113]. We refer to this approach as *the algebraic*



(a) $sparsified(A)$ (b) $sparsified(A^{-1})$

Figure II.4.2: Nonzero pattern for A (left) and A^{-1} (right) when the smallest entries are discarded. The test problem is Example 1.

approach. Several heuristics can be used to define the sparsity pattern based on the magnitude of the entries; all of them result in similar numerical behavior [2] but some are particularly well suited for parallel implementation. In the numerical experiments, we have selected the strategy where, for each column of A, the k entries ($k \ll n$ is a positive integer) of largest modulus are retained.

This strategy generally works well and competes with the approach that adaptively defines the nonzero pattern as implemented in the *SPAI* preconditioner described in [98]. Nevertheless it suffers some drawbacks that put severe limits on its use in practical applications. For large problems, accessing all the entries of the matrix A becomes too expensive or even impossible. This is the case in the fast multipole framework, where all the entries of the matrix A are not available. In addition on complex geometries, a pattern for the sparse approximate inverse computed by using information solely from A may lead to a poor preconditioner. These two main drawbacks motivate the investigation of more appropriate techniques to define a sparsity pattern for the preconditioner.

Because we work in an integral equation context, we can use more information than just the entries of the matrix of the discretized problem. In particular, we can exploit the underlying mesh and extract further relevant information to construct the preconditioner. Two types of information are available from the mesh:

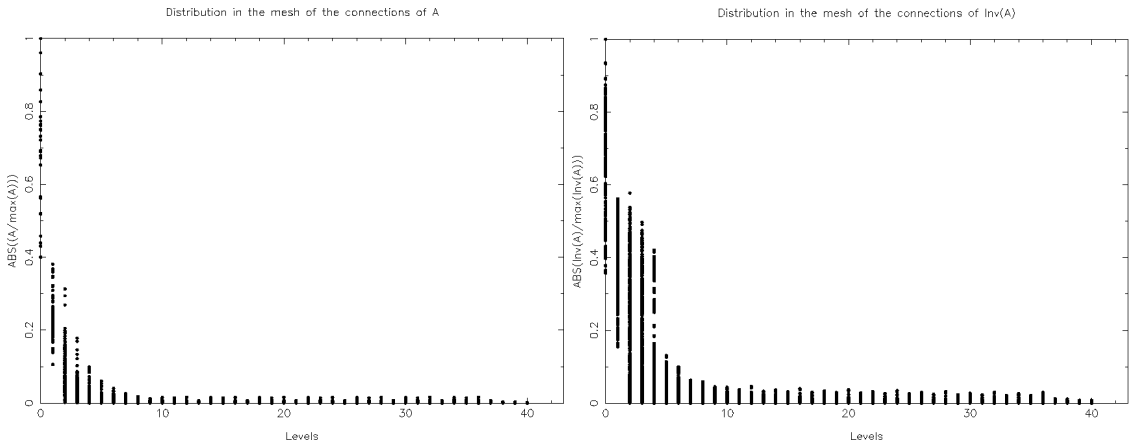
the connectivity graph, describing the topological neighborhood among the edges, and

the coordinates of the nodes in the mesh, describing geometric neighborhoods among the edges.

Topological strategy

When the object geometries are smooth, only the neighboring edges can have a strong interaction with each other, while far-away connections are generally much weaker. Thus an effective pattern for the sparse approximate inverse can be prescribed by exploiting topological information related to the near field. In the integral equation context, the surface of the object is discretized using a triangular mesh. Each degree of freedom (DOF), representing an unknown in the linear system, corresponds to an edge. The sparsity pattern for any row of the preconditioner can be defined according to the concept of level k neighbors, as introduced in [137]. Level 1 neighbors of a DOF

are the DOF plus the four DOFs belonging to the two triangles that share the edge corresponding to the DOF itself. Level 2 neighbors are all the level 1 neighbors plus the DOFs in the triangles that are neighbors of the two triangles considered at level 1, and so forth. In Figure II.4.3 we plot, for each DOF of the mesh for Example 1, the magnitude of the associated entry in A (the graph on the left) and in A^{-1} (the graph on the right) with respect to the level of its neighbors. The large entries in A^{-1} derive from the interaction of a very localized set of edges in the mesh so that by retaining a few levels of neighbors for each DOF an effective preconditioner is likely to be constructed. Three levels can generally provide a good pattern for constructing an effective sparse approximate inverse. Using more levels increases the computational cost but does not improve substantially the quality of the preconditioner. In Figure II.4.4 we show how the density of non-zeros in the preconditioner evolves when the number of level is increased. It can be seen that up-to five levels the preconditioner is still sparse with a density lower than 10 %. We will refer to this pattern selection strategy as the *topological strategy*.



(a) Magnitude v.s. levels for A (b) Magnitude v.s. levels for A^{-1}

FIGURE II.4.3: Topological localization in the mesh for the large entries of A (left) and A^{-1} (right). The test problem is Example 1 and is representative of the general behavior.

Geometric strategy

When the object geometries are not smooth, two far-away edges in the topological sense can have a strong interaction with each other so that they are strongly coupled in the inverse matrix. For the scattering problem on Example 1, we plot in Figure II.4.5, for each pair of edges in the mesh, the magnitude of the associated entry in A (the graph on the left) and A^{-1} (the graph on the right) with respect to their distance in terms of wavelength. The largest entries of A^{-1} are localized similarly to those of A, but, in many cases, small entries in A correspond to large entries in the inverse and vice-versa. This means that if we construct the sparse pattern for the inverse by only using information related to A, we may retain many small entries in the preconditioner, contributing marginally to its quality, but may neglect some of the large ones potentially damaging the quality of the preconditioner. Also, the surface of the object is very non-smooth, these large entries may come from the interaction of far-away or non-connected edges in a topological sense, which are neighbors in a geometric sense. Thus they cannot be detected by using only topological information related to the near field. Figure II.4.5(b) suggests that we can select the pattern for the preconditioner using physical information, that is: for each edge we select all those edges within

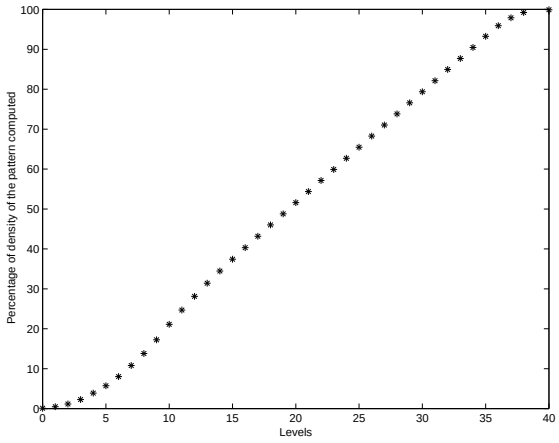
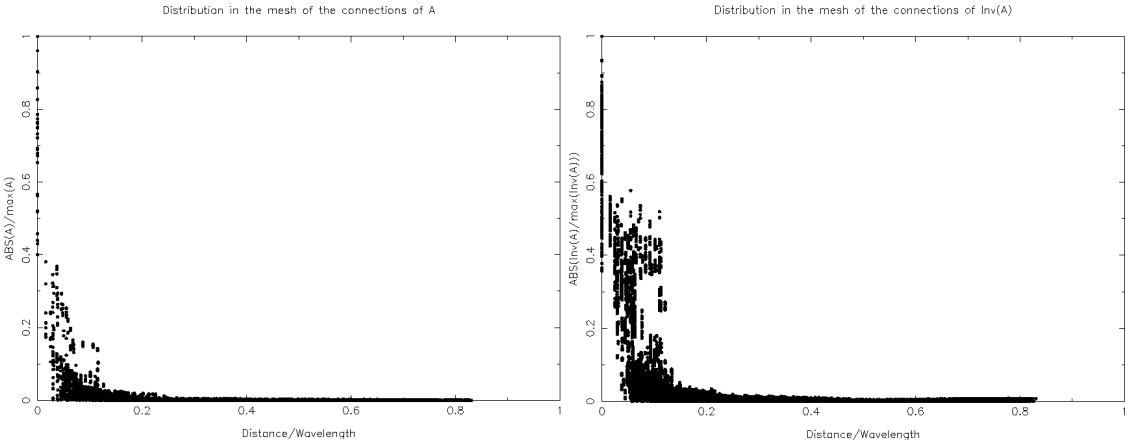


Figure II.4.4: Density of the preconditioner v.s. number of selected levels for Example 1.

a sufficiently large sphere that defines our geometric neighborhood. By using a suitable size for this sphere, we hope to include the most relevant contributions to the inverse and consequently to obtain an effective sparse approximate inverse. In Figure II.4.6 we show how the density of non-zeros in the preconditioner evolves when the radius of the sphere is increased. This selection strategy will be referred to as the *geometric strategy*.



(a) Magnitude v.s. distance for A (b) Magnitude v.s. distance for A^{-1}

Figure II.4.5: Geometric localization in the mesh for the large entries of A (left) and A^{-1} (right). The test problem is Example 1. This is representative of the general behavior.

Numerical experiments

In this section, we compare the different strategies described above in the solution of our test problems.

Using the three pattern selection strategies for M, we denote by

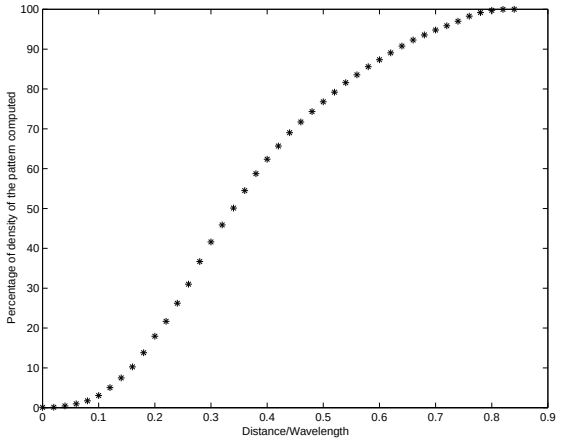


FIGURE II.4.6: Density of the preconditioner v.s. number of selected levels for Example 1.

- M_a, the preconditioner computed by using the algebraic strategy,
- M_t, the preconditioner computed by using the topological strategy,
- M_g, the preconditioner computed by using the geometric strategy,
- $SPAI$, the preconditioner constructed by using the dynamic strategy implemented by [97].

To evaluate the effectiveness of the proposed strategies, we first consider using the dense matrix A to construct the preconditioners M_a, M_t, M_g and $SPAI$. This requires the solution of large dense least-squares problems. The adaptive technique implemented in $SPAI$ computes the pattern of the preconditioner starting with a simple initial guess, like a diagonal matrix, and then improves it until a criterion of the form $\|Am_{j,*} - e_j\|_2 < \varepsilon$ (for each j) is satisfied for a given $\varepsilon > 0$, e_j being the jth column of the identity matrix, and $m_{j,*}$ being the column vector for the jth row of M according to the notation previously introduced, or until a maximum number k of nonzeros in the jth row of M has been generated (we refer the reader to [97] and [98] for further details).

The density of the preconditioner varies from one problem to another for the same value of the distance parameter chosen to define M_g. As Figure II.4.5(b) shows, and tests on all the other examples confirm, those entries corresponding to edges contained within a sphere of radius 0.12 times the wavelength can retain many of the large entries of the inverse while giving rise to quite a sparse preconditioner. For all our numerical experiments, we choose a value for k in the construction of M_a and $SPAI$, and for the level of neighbors used to generate M_t so that they have the same density as M_g, when necessary discarding some small entries of the preconditioner so that all have the same number of entries.

For all the numerical experiments reported in this chapter, for GMRES we use the implementation described in [69]. For the tests with Bi-CGSTAB, we derived a version for complex arithmetic from the Harwell Subroutine Library (HSL, [105]) routine MI06 and for those with unsymmetric QMR (referred to as UQMR in the forthcoming tables) and TFQMR, we used, respectively, the ZUCPL and ZUTFX routines available in QMRPACK [76]. The stopping criteria in all cases just consists in reducing the original residual by 10^{-5}. The symbol "-" means that convergence was not obtained after 500 iterations. In each case, we took as the initial guess $x_0 = 0$, and the right-hand side was such that the exact solution of the system was known. We performed different tests with different known solutions, observing identical results. All the numerical experiments

were performed in double precision complex arithmetic on a SGI Origin 2000 and the number of iterations reported in this chapter are for left preconditioning. Very similar results were obtained when preconditioning from the right.

Example 1 - Density of M = 5.03%								
Precond.	GMRES(m)					Bi - CGStab	UQMR	TFQMR
	m=10	m=30	m=50	m=80	m=110			
Unprec.	-	-	-	251	202	223	231	175
M_j	-	-	465	222	174	239	210	169
M_a	219	135	96	72	72	86	107	72
M_t	100	49	36	36	36	35	42	32
M_g	124	68	46	46	46	44	58	38
SPAI	-	67	44	44	44	48	50	43
Example 2 - Density of M = 1.59%								
Precond.	GMRES(m)					Bi - CGStab	UQMR	TFQMR
	m=10	m=30	m=50	m=80	m=110			
Unprec.	-	-	-	398	289	359	403	249
M_j	-	-	473	330	243	257	354	228
M_a	472	273	239	207	184	330	313	141
M_t	-	470	346	243	195	187	275	158
M_g	90	72	55	52	52	44	82	40
SPAI	-	-	99	61	61	168	97	111
Example 4 - Density of M = 1.04%								
Precond.	GMRES(m)					Bi - CGStab	UQMR	TFQMR
	m=10	m=30	m=50	m=80	m=110			
Unprec.	-	224	191	158	147	177	170	118
M_j	350	211	178	153	140	188	152	110
M_a	212	157	141	132	123	131	145	115
M_t	288	187	160	146	139	145	156	98
M_g	63	51	41	41	41	37	47	32
SPAI	-	370	184	112	84	256	96	85

Table II.4.1: Number of iterations using the preconditioners based on dense A.

From the results shown in Table II.4.1, we first note that all the preconditioners accelerate the convergence of the Krylov solvers, and in some cases enable convergence when the unpreconditioned solver diverges or converges very slowly. These numerical experiments also highlight the advantages of the geometric strategy. It not only outperforms the algebraic approach and is more robust than the topological approach, which has a similar computational complexity, but it also generally outperforms the adaptive approach implemented in SPAI which is much more sophisticated and more expensive in execution time and memory. SPAI competes with M_g only on Example 1 where the density of the preconditioner is higher. This trend, namely the denser the preconditioner the more efficient SPAI is, has been observed on many other examples. However, for sparse preconditioners, SPAI may be quite poor as illustrated on Example 4, where preconditioned GMRES(30) or Bi-CGStab are slower than without a preconditioner and the iteration diverges for

GMRES(10) with the SPAI preconditioner while it converges for the other three preconditioners. On the non-smooth geometry, that is Example 2, an explanation of why the geometric approach should lead to a better sparse preconditioner can be suggested by Figure 4.1(b). Some far-away edges in the connectivity graph, those from each side of the break, are weakly connected in the mesh but can have a strong interaction with each other, and can lead to large entries in the inverse matrix.

4.2.2 Strategies for the coefficient matrix

When the coefficient matrix of the linear system is dense, the construction of even a very sparse preconditioner may become too expensive in execution time as the problem size increases. Both memory and execution time are significantly reduced by replacing A with a sparse approximation. On general problems, this approach can cause a severe deterioration of the quality of the preconditioner; in the BEM context, since a very sparse matrix can retain the most relevant contributions to the singular integrals, it is likely to be more effective. The use of a sparse matrix substantially reduces the size of the least-squares problems that can then be efficiently solved by direct methods.

The algebraic heuristic described in the previous sections is well suited for sparsifying A. In [2] the same nonzero sparsity pattern is selected both for A and M; in that case, especially when the pattern is very sparse, the computed preconditioner may be poor on some geometries. The effect of replacing A with its sparse approximation on some problems is highlighted in Figure II.4.7 where we display the sparsified pattern of the inverse of the sparsified A. We see that the resulting pattern is very different from the sparsified pattern of the inverse of A shown in Figure II.4.2.

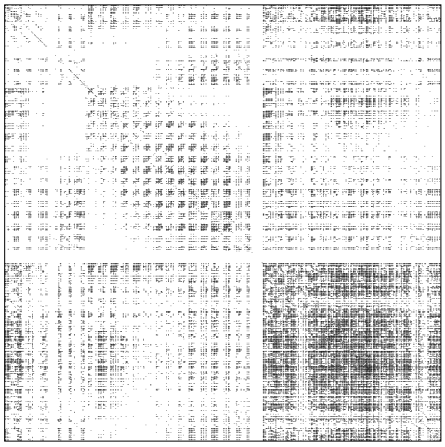


FIGURE II.4.7: Sparsity pattern of the inverse of sparse A associated with Example 1. The pattern has been sparsified with the same value of threshold used to sparsify A displayed in Figure II.4.2.

A possible remedy is to increase the density in the patterns for both A and M. To a certain extent, we can improve the convergence, but the computational cost of generating the preconditioner grows almost cubicly with respect to density. A cheaper remedy is to choose a different number of nonzeros to construct the patterns for A and M, with less entries in the preconditioner than in the sparse approximation of A. To illustrate this effect, we report in Table II.4.2 on the number of iterations of preconditioned GMRES(50), where the preconditioners are built by using either the same sparsity pattern for A or a two, three or five times denser pattern for A. Except

Example 1										
Density strategy	Percentage density of M									
	1	2	3	4	5	6	7	8	9	10
Same	-	-	299	146	68	47	47	42	37	39
2 times	-	-	248	155	76	46	40	39	39	38
3 times	-	253	207	109	49	39	39	37	35	34
5 times	-	258	213	99	48	37	38	34	33	33
Full A	364	359	144	96	46	35	35	34	32	31

Table II.4.2: Number of iterations for GMRES(50) preconditioned with different values for the density of M using the same pattern for A and larger patterns. A geometric approach is adopted to construct the patterns. The test problem is Example 1. This is representative of the general behavior observed.

when the preconditioner is very sparse, increasing the density of the pattern imposed on A for a given density of M accelerates the convergence as expected, getting quite rapidly very close to the number of iterations required when using a full A. The additional cost in terms of CPU time is negligible as can be seen in Figure II.4.8 for experiments on Example 1. This is due to the fact that the complexity of the QR factorization used to solve the least-squares problems is the square of the number of columns times the number of rows. Thus, increasing the number of rows, that is the number of entries of A, does not penalize significantly the construction of the preconditioner. On the other hand, reducing the density of the preconditioner, that is the number of columns in the least-squares problems, can significantly reduce the overall CPU time. Notice that this observation is true for both left and right preconditioning because, according to (II.4.1) and (II.4.2) the smaller dimension of the matrices involved in the least-squares problems always corresponds to the entries of M to be computed, and the larger to the entries of the sparsified matrix from A.

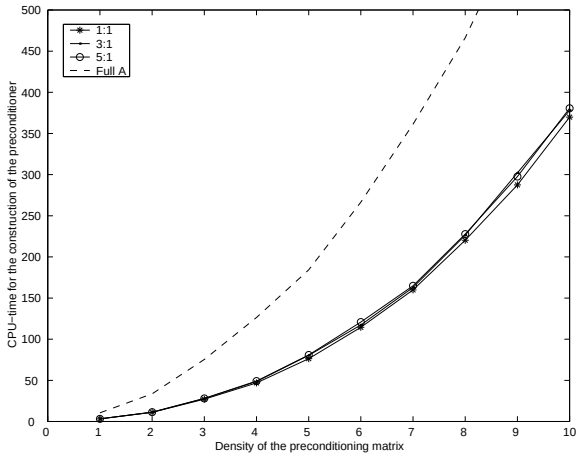


Figure II.4.8: CPU time for the construction of the preconditioner using a different number of nonzeros in the patterns for A and M. The test problem is Example 1. This is representative of the other examples.

Numerical experiments

We report in this section on the numerical results obtained by replacing A with its sparse approximation in the construction of the preconditioner. In Table 3 we use the following notation:

- M_{a-a}, introduced in [2] and computed by using algebraic information from A. The same pattern is used for the preconditioner;
- M_{a-t}, constructed by using the algebraic strategy to sparsify A and the topological strategy to prescribe the pattern for the preconditioner;
- M_{a-g}, constructed by using the geometric approach and an algebraic heuristic for A with the same density as for the preconditioner;
- M_{2a-t}, similar to M_{a-t}, but the density of the pattern imposed on A is twice as dense as that imposed M_{a-t};
- M_{2a-g}, similar to M_{a-g} but, as in the previous case, the density of the pattern imposed on A is twice as dense as that imposed on M_{a-g}.

For the sake of comparison we also report the number of iterations without using a preconditioner and with only a diagonal scaling, denoted by M_j.

Other combinations are possible for defining the selection strategies for the patterns of A and M. Here we focus on the most promising ones, that use information from the mesh to retain the large entries of the inverse, and the algebraic strategy for A to capture the most relevant contributions to the singular integrals. We also consider the preconditioner M_{a-a} to compare with previous tests [2], that were performed on different geometries from those considered here. Although sparsifying A using an algebraic dropping strategy seems to be the most natural approach to get a sparse approximation of A when all its entries are available, either the topological or the geometric criterion can be used to define the sparse approximation of A. Those alternatives are attractive in a multipole framework where all the entries of A are not computed and some results using these strategies are reported in Section 4.3. We show, in Table II.4.4, the results of our numerical experiments. For each example, we give the number of iterations required by each preconditioned solver.

Example 1 - Density of M = 5.03%								
Precond.	GMRES(m)					Bi - CGStab	UQMR	TFQMR
	m=10	m=30	m=50	m=80	m=110			
Unprec.	-	-	-	251	202	223	231	175
M_j	-	-	465	222	174	239	210	169
M_{a-a}	284	170	138	114	92	120	156	94
M_{a-t}	179	61	45	45	45	43	58	36
M_{a-g}	147	93	68	59	59	55	73	53
M_{2a-t}	128	56	40	40	40	37	50	36
M_{2a-g}	131	79	52	51	51	59	65	44

Continued on next page

Continued from previous page

Example 2 - Density of M = 1.59%								
Precond.	GMRES(m)					Bi - CGStab	UQMR	TFQMR
	m=10	m=30	m=50	m=80	m=110			
Unprec.	-	-	-	398	289	359	403	249
M_j	-	-	473	330	243	257	354	228
M_{a-a}	-	319	255	221	203	181	319	135
M_{a-t}	-	261	213	174	169	128	251	121
M_{a-g}	251	178	150	138	117	106	256	116
M_{2a-t}	-	370	284	202	182	176	276	127
M_{2a-g}	100	73	61	55	55	48	93	40
Example 3 - Density of M = 2.35%								
Precond.	GMRES(m)					Bi - CGStab	UQMR	TFQMR
	m=10	m=30	m=50	m=80	m=110			
Unprec.	-	-	-	-	488	-	444	308
M_j	-	-	-	491	427	375	356	306
M_{a-a}	436	316	240	193	125	144	166	135
M_{a-t}	137	108	93	71	71	64	93	66
M_{a-g}	-	464	296	203	108	240	166	144
M_{2a-t}	113	78	59	53	53	41	61	44
M_{2a-g}	122	84	72	59	59	53	67	50
Example 4 - Density of M = 1.04%								
Precond.	GMRES(m)					Bi - CGStab	UQMR	TFQMR
	m=10	m=30	m=50	m=80	m=110			
Unprec.	-	224	191	158	147	177	170	118
M_j	350	211	178	153	140	188	152	110
M_{a-a}	299	205	172	146	133	162	180	103
M_{a-t}	266	152	130	114	99	92	127	83
M_{a-g}	81	67	66	63	63	39	79	41
M_{2a-t}	269	167	143	136	116	107	137	93
M_{2a-g}	71	60	47	47	47	43	61	41
Example 5 - Density of M = 0.63%								
Precond.	GMRES(m)					Bi - CGStab	UQMR	TFQMR
	m=10	m=30	m=50	m=80	m=110			
Unprec.	-	344	233	146	125	152	170	109
M_j	-	326	219	140	131	183	173	107
M_{a-a}	-	352	249	154	134	202	183	107
M_{a-t}	360	66	64	60	60	34	76	46
M_{a-g}	313	81	68	61	61	36	74	40
M_{2a-t}	71	48	47	47	47	25	54	30
M_{2a-g}	88	42	39	39	39	21	45	25

Table II.4.4: Number of iterations to solve the set of test problems.

Example 1 - Density of $M = 5.03\%$				
M_{a-a}	M_{a-t}	M_{2a-t}	M_{a-g}	M_{2a-g}
83.42	91.07	91.78	79.47	80.18

Example 2 - Density of $M = 1.59\%$				
M_{a-a}	M_{a-t}	M_{2a-t}	M_{a-g}	M_{2a-g}
13.98	16.45	16.73	13.53	13.67

Example 3 - Density of $M = 2.35\%$				
M_{a-a}	M_{a-t}	M_{2a-t}	M_{a-g}	M_{2a-g}
83.59	146.44	147.79	109.45	110.30

Example 4 - Density of $M = 1.04\%$				
M_{a-a}	M_{a-t}	M_{2a-t}	M_{a-g}	M_{2a-g}
31.75	38.05	38.23	31.12	31.24

Example 4 - Density of $M = 0.63\%$				
M_{a-a}	M_{a-t}	M_{2a-t}	M_{a-g}	M_{2a-g}
27.66	70.93	71.29	26.04	26.13

TABLE II.4.5: CPU time to compute the preconditioners.

In Table II.4.5 we report the CPU time required to compute the preconditioners when the least-squares problems are solved using LAPACK routines. The CPU time for constructing M_{a-t} and M_{2a-t} is in some cases much larger than that needed for M_{a-g} and M_{2a-g}. The reason is that, in the topological strategy, it is not possible to prescribe exactly a value for the density. Thus, for each problem, we select a suitable number of levels of neighbors, to obtain the closest number of nonzeros to that retained in the pattern based on the geometric approach. After the construction of the preconditioner, we drop its smallest entries to ensure an identical number of nonzeros for the two strategies. The results illustrate that considering twice as dense a pattern for A as for M does not cause a significant growth in the computational time although it enables us to construct a more robust preconditioner.

We first observe that using a sparse approximation of A reduces the convergence rate of the preconditioned iterations when the nonzero pattern imposed on the preconditioner is very sparse. However if we adopt the geometric strategy to define the sparsity pattern for the approximate inverse, the convergence rate is not affected very much. For even larger values of density, the difference in the number of iterations between using full A or an algebraic sparse approximation becomes negligible. For all the experiments, M_{a-g} still outperforms M_{a-a} and is generally more robust than M_{a-t}; the most efficient and robust preconditioner is M_{2a-g}. The multiple density strategy allows us to improve the efficiency and the robustness of the Frobenius-norm preconditioner on this class of problems without requiring any more time for the construction of the preconditioner. For all the test examples, it enables us to get the fastest convergence even for GMRES with a low restart parameter on problems where neither M_{a-a} nor M_{a-g} converge.

The effectiveness of this multiple density heuristic is illustrated in Figure II.4.9 where we see the effect of preconditioning on the clustering of the eigenvalues of A for the most difficult problem, Example 2. The eigenvalues of the preconditioned matrices are in both cases well clustered around 1 (with a more effective clustering for M_{2a-g}), but those obtained by using the multiple density strategy are further from the origin. This is highly desirable when trying improve the convergence of Krylov solvers.

Another advantage of this multiple density heuristic is that it generally allows us to reduce the density of the preconditioner (and thus its construction cost), while preserving its numerical quality. Although no specific results are reported to illustrate this aspect, this behavior may be

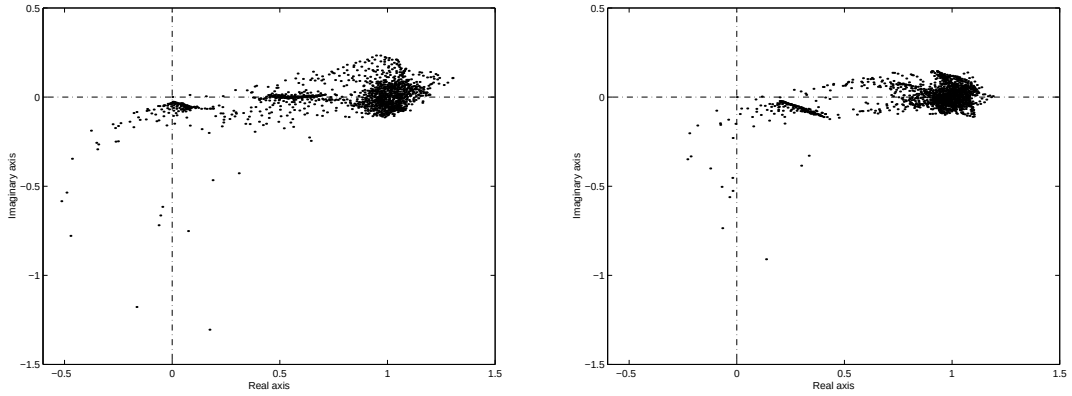


FIGURE II.4.9: Eigenvalue distribution for the coefficient matrix preconditioned by using a single (on the left) and a multiple (on the right) density strategy on Example 2.

partially observed in Table II.4.2.

Example 1 - Density of $M = 5.03\%$								
Precond.	GMRES(m)					Bi - CGStab	UQMR	TFQMR
	m=10	m=30	m=50	m=80	m=110			
M_j	-	-	465	222	174	239	210	169
$SSOR$	-	-	216	136	98	147	177	135
$ILU(0)$	-	-	-	-	-	-	479	-
$SPAI$	-	-	192	68	68	150	83	94
SLU	160	53	38	38	38	46	50	39
M_{2a-g}	131	79	52	51	51	59	65	44
Example 2 - Density of $M = 1.59\%$								
Precond.	GMRES(m)					Bi - CGStab	UQMR	TFQMR
	m=10	m=30	m=50	m=80	m=110			
M_j	-	-	473	330	243	257	354	228
$SSOR$	-	413	245	164	134	185	281	266
$ILU(0)$	-	-	-	-	322	385	394	439
$SPAI$	-	-	-	-	-	-	-	-
SLU	-	-	-	-	282	-	-	-
M_{2a-g}	100	73	61	55	55	48	93	40

TABLE II.4.6: Number of iterations with some classical preconditioners computed using sparse A (algebraic).

Finally, to assess the performance of the proposed Frobenius-norm minimization approach described in this chapter, we show, in Table II.4.6, the numerical results observed on Examples 1 and 2 with some classical preconditioners, of both explicit and implicit form. These are: diagonal scaling, SSOR, ILU(0) and SPAI applied to a sparse approximation of A constructed using the algebraic approach. The method referred to as SLU in that table uses the sparsified matrix A

as an implicit preconditioner; that is, the sparsified matrix is factorized using ME47, a sparse direct solver from HSL, and those exact factors are used as the preconditioner. Thus it represents an extreme case with respect to $ILU(0)$, since a complete fill-in is allowed in the factors. This approach, although not easily parallelizable, is generally quite effective on this class of applications for dense enough sparse approximations of A. However, as shown in this table, when the preconditioner is very sparse, the numerical quality of this approach deteriorates and the Frobenius-norm minimization method is more robust. All these preconditioners, except SLU on Example 1, exhibit much poorer acceleration capabilities than that provided by M_{2a-g}. If we reduce the density of the preconditioner in Example 1, M_{2a-g} converges slowly but becomes the most efficient. It should also be noted that SPAI works reasonably well when computed using dense A (see Table II.4.1) but with sparse A it does not converge on Example 2 (see Table II.4.6). In addition, following [49], we performed some numerical experiments where we obtained an approximate $m_{*,j}$ from (II.4.2) by dropping the smallest entries of the iterates computed by few steps of either the Minimum Residual method or GMRES. Unfortunately, the performance of these approaches for dynamically defining the pattern of the preconditioner was disappointing. They only improved the unpreconditioned case when a relative large number of iterations was used to build the preconditioner making them unaffordable for our problems.

Our purpose in this chapter is to study the numerical behavior of the preconditioners. Nevertheless, we do recognize that some of the simple strategies have a much lower cost to build the preconditioner and so could result in a faster solution. When SSOR converges, it is often the fastest, in terms of the overall CPU time for the overall solution of the linear system. When the solution is performed for only one right-hand side, the construction cost of the other preconditioners cannot be compensated for by the reduction in the number of iterations; the matrix-vector product is performed using BLAS kernels that make the iteration cost quite cheap for the problem sizes we have considered. For instance, when solving Example 1 with GMRES(50) on a SUN Enterprise, SSOR converges in 31.4 seconds and M_{2a-g} requires 190 seconds for the construction and 7.6 seconds for the iterations. However, in electromagnetism applications, the same linear system has to be solved with many right-hand sides when illuminating an object with various waves corresponding to different angles of incidence. For that example, if we have more than eight right-hand sides, the construction of M_{2a-g} is overcome by the time saved in the iterations and M_{2a-g} becomes more efficient than SSOR. In addition, the construction and the application of M_{2a-g} is fully parallelizable while the parallelization of SSOR requires some reordering of equations that may be difficult to implement efficiently on a distributed memory platform.

4.3 Concluding remarks

We have presented some *a priori* pattern selection strategies for the construction of a robust sparse Frobenius-norm minimization preconditioner for electromagnetic scattering problems expressed in integral formulation. We have shown that, by using additional geometric information from the underlying mesh, it is possible to construct robust sparse preconditioners at an affordable computational and memory cost. The topological strategy requires less computational effort to construct the pattern, but since the density is a step function of the number of levels, the construction of the preconditioner can require some additional computation. Also it may not handle very well complex geometries where some parts of the object are not connected, as in Example 3 (see Figure 4.1(c)). By retaining two different densities in the patterns of A and M we can decrease very much the computational cost for the construction of the preconditioner, usually a bottleneck for this family of methods; preserving the efficiency while increasing the robustness of the resulting preconditioner. The numerical experiments have shown that, using this pattern selection strategy, we can compute a very sparse but effective preconditioner. With the same low density, none of

the classical preconditioners that we considered can compete with it. An additional major feature of this pattern selection strategy is that it does not require access to all the entries of the matrix A, so that it is promising for an implementation in a fast multipole setting where A is not directly available but where only the near field entries are computed.

M_{2g-g}								
Example	GMRES(m)					Bi - CGStab	UQMR	TFQMR
	m=10	m=30	m=50	m=80	m=110			
1	165	103	75	60	60	66	71	61
2	145	110	95	76	76	68	140	64
3	129	89	70	57	57	49	69	52
4	71	57	48	48	48	38	52	34
5	110	46	42	42	42	24	50	27

Table II.4.7: Number of iterations to solve the set of test models by using a multiple density geometric strategy to construct the preconditioner. The pattern imposed on M is twice as dense as that imposed on A.

M_{2t-g}								
Example	GMRES(m)					Bi - CGStab	UQMR	TFQMR
	m=10	m=30	m=50	m=80	m=110			
1	197	87	49	49	49	50	66	50
2	103	82	72	61	61	49	111	50
3	143	98	84	60	60	56	70	53
4	70	58	49	49	49	39	65	37
5	143	50	47	47	47	29	57	28

Table II.4.8: Number of iterations to solve the set of test models by using a topological strategy to sparsify A and a geometric strategy for the preconditioner. The pattern imposed on M is twice as dense as that imposed on A.

The geometric approach can be also used to sparsify A, without noticeably deteriorating the quality of the preconditioner. This is showed in Table II.4.7, where M_{2g-g} is constructed by exploiting geometric information in the patterns of both A and M, but choosing twice as dense a pattern for A as for M. As suggested by Figure II.4.3(a), due to the strongly localized coupling introduced by the discretization of the integral equations, the topological approach can also provide a good sparse approximation of A, by retaining just a few levels of neighboring edges for each DOF in the mesh. The numerical behavior of this approach is illustrated in Table II.4.8. In both cases the resulting preconditioner is still robust and better suited for a fast multipole framework since it does not require knowledge of the location of the largest entries in A.

Part III

Parallel performance of partial differential equation solvers

Chapter 1

Parallel performance of asynchronous iterations

1.1 Introduction

Asynchronous iterations by essence remove most of the synchronization points that usually are the critical aspects where particular attention should be paid on when implementing algorithms on parallel computers. In consequence their parallel implementation both on shared and distributed memory computers are straightforward. On a distributed memory computer this observation is only true if asynchronous communication are supported by the message passing library available on the target platform.

Unfortunately at the time the numerical experiments reported in this chapter were performed no standard like PVM [11] or MPI [124] for message passing existed. Even worse the target parallel distributed platform was a network of Transputer that only supported the OCCAM language [59]. This parallel computing environment had strong impact on the parallel implementation design of asynchronous algorithms because:

1. the OCCAM language only implements synchronous communication through blocking send and receive, based on rendez-vous C.S.P. [103],
2. the Transputer chip is composed by a processor plus its memory for the computation and four links to be physically connected to at most four other Transputer enabling a Transputer to have point to point communication with only its four neighbors. This means that a Transputer network is not a fully connected distributed architecture and consequently the topology of the network should be tuned to the particular algorithm to be implemented.

The main consequences on the design of asynchronous algorithms are:

1. an additional layer of communication protocol should be implemented to enable asynchronous communication among processors that perform the relaxation iterations,
2. only 1D decomposition (strips) of the 2D domain can be considered, since box decomposition would have required more than four links per Transputer when using our asynchronous communication protocol or would have required to use more Transputer for managing communication than for the actual computation.

This chapter is organized as follows. In Section 1.2 we describe the implementation of asynchronous iterations on a Transputer network and report in Section 1.3 comparisons of performance

between parallel synchronous and asynchronous SOR iterations for the solution of the discretized and linearized Hamilton-Jacobi-Bellman equations (II.1.7).

1.2 An implementation of asynchronous iterations on a Transputer network

To implement asynchronous linear relaxations using the blocking communication available in the OCCAM language, we consider a ring topology and split the set of Transputer in two subsets. One subset performs the actual computation, the other subset manages the communication. Asynchronous relaxations are then performed in such a way that each computing Transputer solves a local problem for which the boundary conditions on the interfaces with its neighbors change asynchronously while the iterations progress.

For 2D meshes this approach only enables to consider strip decompositions and the mapping of the non-overlapping subdomains on the Transputer ring is depicted in Figure III.1.1 for a decomposition of the physical domain Ω into three subdomains $(\Omega_1, \Omega_2, \Omega_3)$; (T_1, T_3, T_5) is the subset of computing nodes, (T_2, T_4) is the subset of Transputer that manage the communication and finally T_6 is in charge of monitoring the convergence as it will be described later on. The commu-

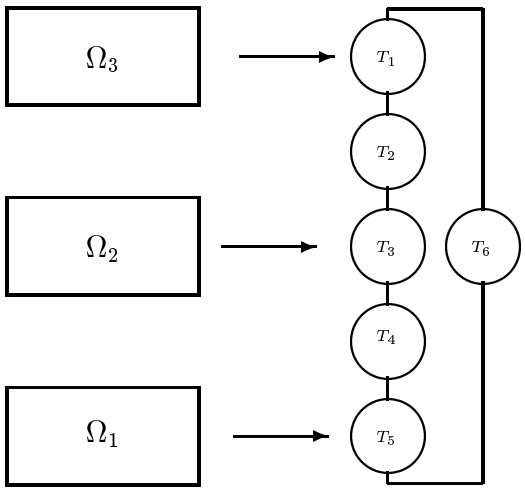


FIGURE III.1.1: Ring topology and mapping of the subdomains on the Transputer.

nication protocol that has been implemented is such that computing Transputer send read/write requests to their neighboring communication Transputer to get updated boundary conditions on the corresponding interface or to provide their neighbors with up-to-date boundary conditions. The asynchronous behavior is obtained thanks to a feature of the OCCAM language that enables to wait simultaneously for several messages and as soon as a message has arrived a specific task can be started. This feature is very similar to the interruptions management using handlers on microprocessors. Then handling the first received requests enables to simulate asynchronous communication between two computing Transputer using blocking send and receive, the price to pay is to only have half of the computing resource actually used for real computation. The last Transputer, T_6 in Figure III.1.1 that closes the loop, only performs the evaluation of the stopping criterion using a token that circulates in a prescribed order among the Transputer. The token is composed by two slots,

a boolean describing the convergence status that informs the computing Transputer that they have to stop to iterate,

and a real enabling to evaluate the 2-norm of the residual.

The token evolves through the ring each time a read or write request is made by the computing Transputer to the communication node that holds the token. The slot containing the real value is updated by the computing Transputer that accumulate in this variable the 2-norm of the residual computed locally on the sub-domain it is in charge of. In this way when the token comes back to the Transputer evaluating the stopping criterion, the real slot contains an estimation of the 2-norm of the overall residual.

Using strip decomposition and lexicographical ordering of the finite difference grid, the classical SOR relaxations can be parallelized using a pipelined approach. The parallel algorithm starts with only the processor working on Ω_1 that performs its first relaxation on its points; once completed, it sends the values of the first iterates on its interface with Ω_2 to the processor in charge of Ω_2 that may starts its first iteration while processor one performs its second iterations, and so one and so forth for the other processors. At a given time all the processors compute what are the iterates produced by a sequential SOR method but each processor have the iterates at a different iteration step. The SPMD pseudo code of this parallel synchronous block SOR implementation can be described as follows:

```
      nbiter = 0
      REPEAT
*
          recvfrom(iproc-1)
          Relax first discretization line
          sendto(iproc-1)
*
          Relax other discretization lines except the last
*
          if (nbiter.ne.0) then
             recvfrom(iproc+1)
             Relax last discretization line
          endif
*
          sendto(iproc+1)
     UNTIL convergence
```

In the sequel this parallel implementation is referred to as the synchronous parallel SOR method.

1.3 A comparison of synchronous and asynchronous parallel iterations

In this section we compare the behavior of parallel asynchronous iterations versus parallel synchronous iterations on the Hamilton-Jacobi-Bellman problem described in the previous part. In Table III.1.1 we report the best observed performance on a network on Transputer for which both the partitioning and the over-relaxation parameter were tuned. The speed-up is classically computed as the ratio of the sequential elapsed time divided by the parallel elapsed time. The domain is discretized using a uniform and regular 143×143 grid. The super-linear speed-ups observed on

# subdomains	mode	min/Max iterations	Speed-up
1	sequential	241	1.00
2	synchronous	241	2.07
	asynchronous	244/247	2.05
4	synchronous	241	4.10
	asynchronous	241/291	3.88
8	synchronous	241	7.89
	asynchronous	145/239	10.89

TABLE III.1.1: Performance observed on a Transputer network.

two subdomains and with four subdomains with the synchronous method are due to cache effects. But the super-linear speed-up with eight subdomains and asynchronous iterations is introduced by the smaller iterations required by this method with respect to the sequential execution.

Both the synchronous and asynchronous parallel algorithms described in the previous section can be implemented on a shared memory multiprocessor using only simple parallelization directives. In Table III.1.2, we depict performance observed on a Alliant FX/80, that we used in dedicated mode to run those experiments.

# subdomains	mode	min/Max iterations	Speed-up
1	sequential	241	1.00
2	synchronous	241	1.95
	asynchronous	255/258	1.83
4	synchronous	241	3.64
	asynchronous	238/271	3.51
8	synchronous	241	6.39
	asynchronous	144/228	9.05

TABLE III.1.2: Performance observed on a Alliant FX/80.

These results show that the asynchronous iterations are potentially as fast and sometimes even faster that the corresponding synchronous implementations. However their numerical behavior is very difficult and complex to predict; it changes from one computer to another and even sometimes two successive executions in a dedicated mode give different convergence behavior. In particular the optimal over-relaxation parameter on the Alliant and on the Transputer network for a given decomposition are different and changes when the number of subdomains is changed.

1.4 Concluding remarks

Extensive theoretical works have been done to analyze the convergence of these methods and recently new asynchronous schemes with "flexible communication" have been introduced [130] to enable more asynchronism in block algorithms. Nevertheless a challenging study targeting a better understanding and prediction of their parallel numerical behavior would deserve to be developed to make these methods more reliable for an usage in large simulation codes. For parallel linear algebra solvers, it can also be though to use these asynchronous relaxation iterations as smoother in multigrid or to consider few steps of asynchronous relaxations as a preconditioner of FGMRES [140] or

GMRESR [153], that are Krylov solvers in which it is allowed to take a different preconditioner in each step.

Chapter 2

Parallel performance of two level non-overlapping domain decomposition methods

2.1 Introduction

The full exploitation of the new computer architecture with large processor numbers requires parallel scalable implementations of numerically scalable numerical techniques. The numerical scalability of a numerical algorithm is characterized by the independence of its numerical behavior with respect to the problem size it enables to solve. For elliptic partial differential equations solution, multigrid on smooth problems is numerically scalable [100] and for general matrices clearly direct methods are (when they are affordable). Multi-level domain decomposition methods quasi satisfy this criterion in the sense that their numerical behavior is independent from the number of subdomains and only weakly depends on the subdomain size. On the other hand, parallel scalable implementations are characterized by quasi constant elapsed time for performing one step of the numerical algorithm when the overall problem size is increased linearly with the number of processors used. Straightforward parallel implementations of explicit schemes for the solution of time dependent problems, Jacobi/Richardson stationary methods and unpreconditioned Krylov method for solving discretized problem are scalable (some attention should be paid to perform the scalar products on large processor number in this later case, see for instance [125, 53, 71]). Variants of multigrid can give rise to scalable parallel implementation and multi-level domain decomposition technique are also good candidate for parallel scalable implementations. In this chapter, we describe the implementation of a parallel scalable two-level Schur complement domain decomposition. In Section 2.2 the parallel implementation of some of two-level preconditioners we have described in Part II Chapter 2 is presented. The parallel performance observed on a Cray T3D is reported in Section 2.3.

2.2 Parallel implementation

The independent solution of local PDE problems expressed by the domain decomposition techniques are particularly suitable for parallel distributed computation. In a parallel distributed memory environment each subdomain can be assigned to a different processor. With this mapping, all the basic linear algebra operations but two in the preconditioned conjugate gradient can

be implemented either without or with only neighbor-to-neighbor communication. The only two steps that require global communication are the dot product computation and the solution of the coarse problem performed at each iteration. In the sequel, we will describe how the preconditioner composed by edge multicolor probed as local preconditioner and any of the coarse component described in Section 2.3 can be efficiently implemented on distributed memory platforms. We will also describe how the coarse problem solution can be implemented without any extra global communication within the iteration loop.

A time consuming kernel involved at each step of the preconditioned conjugate gradient is the matrix vector product when the Schur complement is not built explicitly, that is when only the local Dirichlet matrices are factorized and forward/backward substitution steps are required at each iterations. To perform the Cholesky factorization we have performed experiments with several direct solvers. Form the simplest to the more sophisticated we have considered the band solver from LAPACK [6], a skyline solver [79] and MA27 [62] from Harwell Subroutine Library [105].

The results observed on one processor of the Cray T3D and reported in Table III.2.1 show the clear superiority of MA27 with respect to the other two solvers both in term of memory requirements and in term of computational time.

solver	Memory requirement (in Mbytes)	Factorization times (in sec)	Solve times (in sec)
Band	2.12	0.875	0.103
Skyline	1.31	1.034	0.061
MA27	0.70	0.409	0.027

TABLE III.2.1: Performance of the different linear solvers on one 64×64 subdomain.

In the sequel we will only report experiments with MA27 as local direct solver.

2.2.1 Parallel edge probing implementation

A straightforward implementation of the multicoloring probing technique is to perform a sequence of matrix-vector product between the implicit Schur complement matrix and each set of $(2d+1)$ probing vectors associated with each interface of the subdomains. This approach involved neighbor to neighbor communication between processors for each of the probing vectors on the shared interface and is referred to as the matrix-vector approach. In order to reduce the number of communication, the communication involved for each matrix-vector product can be postponed until each processor has completed the computation of its local Schur complement times all the probing vectors defined on its interface. With this approach only one communication between neighbors is required to communicate a matrix which columns are composed by the local matrix-probing vector results. This second approach is referred to as the matrix-matrix implementation. In both cases the amount of exchanged data is the same, but the second one minimize the network latency overhead since only one communication is performed. In counterpart the price to pay is to store all the partial vector resulting from the local Schur complement matrix applied to the set of probing vectors.

The advantage of the matrix-matrix implementation is illustrated in Table III.2.2 where we report both the synchronization and communication elapsed time in the column "comm", the total elapsed time to construct the probed edge local preconditioner and the percentage of the communication/synchronization with respect to the overall construction.

	matrix-matrix			matrix-vector		
#pprocessors	coma.	total	%	comm.	total	%
2×2	32	235	13.4	42	256	16.4
2×4	62	319	19.4	111	382	29.0
4×4	67	373	17.9	163	484	33.6
4×8	51	390	13.0	138	486	28.4

Table III.2.2: Elapsed time in milliseconds for communication/synchronization and computation during the construction of a preconditioner using matrix-matrix and matrix-vector approaches, each subdomain is a 64×64 grid.

2.2.2 Parallel coarse component construction and application

The linear systems associated with the coarse spaces are much smaller than the linear systems associated with the local Dirichlet problems, which have to be solved when computing the matrix vector product by S. In this respect, we construct the coarse matrix A_0 once and assemble it on all the processors so that we can redundantly perform in parallel its solution at each preconditioned conjugate gradient iteration. Furthermore, we can take advantage of the structure of S and R_0 to construct A_0 in parallel. Without any communication each processor can compute the contribution of its subdomain to the entries of A_0 via matrix-vector and scalar products that only involve its local Schur complement and the vectors whose support intercept the boundary of its subdomain. At this stage, all the processors have some non-assembled entries of A_0, a global sum reduction (MPI_Allreduce) enables them to assemble A_0 on all the processors that can then factorize it.

As the Schur complement matrix is not assembled, the most expensive part of the construction is the matrix vector product with the local Schur complement that requires the solution of the Dirichlet problems. For each processor, the number of solutions is equal to the number of basis vector supports that intercept the boundary of the subdomain the processor is in charge of. For a box-decomposition of a uniform finite elements or finite differences mesh, the number of Dirichlet problem solutions to be performed by an internal subdomain is:

- four for the vertex-based coarse-component,
- eight for the subdomain based coarse-component (that can reduce to four for a five point finite difference scheme as the row in A associated to the cross points is unchanged in S),
- four for the edge based coarse-component.

Having made the choice of a redundant solution of the coarse component on each processor, we can exploit further this formulation to avoid introducing any new global synchronization in the

preconditioned conjugate gradient (PCG) iterations described below.

$$
\begin{aligned}
& x^{(0)} = 0,\ r^{(0)} = b \\
& \textbf{repeat} \\
& \quad z^{(k-1)} = M r^{(k-1)} \qquad \text{(III.2.1)} \\
& \quad \textbf{if } k = 1 \textbf{ then} \\
& \quad\quad p^{(1)} = z^{(0)} \\
& \quad \textbf{else} \\
& \quad\quad \beta^{(k-1)} = \boxed{z^{(k-1)^T} r^{(k-1)}} / z^{(k-2)^T} r^{(k-2)} \qquad \text{(III.2.2)} \\
& \quad\quad p^{(k)} = z^{(k-1)} + \beta^{(k-1)} p^{(k-1)} \qquad \text{(III.2.3)} \\
& \quad \textbf{endif} \\
& \quad q^{(k)} = S p^{(k)} \\
& \quad \alpha^{(k)} = z^{(k-1)^T} r^{(k-1)} / \boxed{p^{(k)^T} q^{(k)}} \\
& \quad x^{(k)} = x^{(k-1)} + \alpha^{(k)} p^{(k)} \\
& \quad r^{(k)} = r^{(k-1)} - \alpha^{(k)} q^{(k)} \\
& \textbf{until convergence}
\end{aligned}
$$

The steps involving a potential global synchronization are boxed, while the calculation of Mr and Sp only involve neighbor to neighbor communication.

If we now unroll Equation (III.2.1) in the PCG algorithm using the general definition of the preconditioner, we have

$$
\begin{aligned}
z_k &= (\sum_{E_i} R_i^T \tilde{S}_{ii}^{-1} R_i + R_0^T A_0^{-1} R_0) r_k \\
&= \sum_{E_i} (R_i^T \tilde{S}_{ii}^{-1} R_i r_k) + R_0^T A_0^{-1} R_0 r_k. \qquad \text{(III.2.4)}
\end{aligned}
$$

Each term of the summation in Equation (III.2.4) is computed by one processor with possible one neighbor-to-neighbor communication. Furthermore, the numerator of Equation (III.2.2) can also be rewritten as

$$
\begin{aligned}
(r_k, z_k) &= (r_k, M r_k) \\
&= (r_k, (\sum_{E_i} R_i^T \tilde{S}_{ii}^{-1} R_i + R_0^T A_0^{-1} R_0) r_k) \\
&= \sum_{E_i} (R_i r_k, \tilde{S}_{ii}^{-1} R_i r_k) + (R_0 r_k, A_0^{-1} R_0 r_k). \qquad \text{(III.2.5)}
\end{aligned}
$$

The right-hand side of (III.2.5) has two parts. The first is naturally local because it is related to the diagonal block preconditioner. The second, with the presented formulation, is global but does not require any new global reduction. $R_0 r_k$ is actually composed of entries that are calculated in each subdomain ("interface" coarse space) or group of neighboring subdomains ("vertex" and "domain" coarse spaces). After being locally computed, the $R_0 r_k$ entries are gathered on all the processors thanks to the reduction used to assemble each local partial dot product $(R_i r_k, \tilde{S}_{ii}^{-1} R_i r_k)$. At this stage the solution $A_0^{-1} R_0 r_k$ can be performed redundantly on each processor as well as β in Equation (III.2.2) can be computed by each processor.

Rewriting these steps in the iteration loop allows us to introduce the coarse component without

any extra global synchronization. With this approach, we avoid a well-known bottleneck when using Krylov methods on parallel distributed memory computers.

2.3 Parallel experiments

We investigate the parallel scalability of the proposed implementation of the preconditioners. For each experiment, we map one subdomain on each processor of the parallel computer. In the sequel, the number of subdomains and the number of processors will be always the same. The target computer is a 128-node T3D located at CERFACS, using MPI as message passing library. The factorization of the tridiagonal probed matrices, used in the local part of the preconditioners is performed using a LAPACK [6] band solver. For all the experimental results reported in the next section, the convergence of the preconditioned conjugate gradient method is attained when the 2-norm of the residual of the current iteration normalized by the 2-norm of the right hand side is less than 10^{-6}, the initial guess x_0 for the conjugate gradient iterations was the null vector. All the experiments were performed in double precision arithmetic.

To study the parallel behavior of the code, we report the maximum elapsed time (in seconds) spent by one of the processors in each of the main steps of the domain decomposition method when the number of processors is varied for solving the standard Poisson's equation. The first row, entitled "init.", corresponds mainly to the time for factorizing the matrix associated with the local Dirichlet problems; "setup local" is the time to construct and factorize the probing approximations $\tilde{S}_{ii}$; "setup coarse" is the time required to construct and factorize the matrix associated with the coarse problem; "iter" is the time spent in the iteration loop of the preconditioned conjugate gradient. Finally, the row "total" permits to evaluate the parallel scalability of the complete methods (i.e. numerical behavior and parallel implementation), while "time per iter." only illustrates the scalability of the parallel implementation of the preconditioned conjugate gradient iterations. The elapsed time corresponds to a maximum and there is some unbalance among the processors in different kernels. Therefore the reported total time differs from the sum of the time for each individual kernel.

To illustrate the extra cost introduced by the construction and the solution of the coarse problem at each iteration, we give in Tables III.2.3 and III.2.4 the time spent in each step of the algorithm with (left column) and without (right column) the considered coarse component of the preconditioner.

We report experiments with the domain-based coarse space in Table III.2.3. Results with the vertex-based coarse space are displayed in Table III.2.4. For those experiments, we use MA27 to solve the local Dirichlet problems defined on 100×100 grids. That subdomain size was the largest we could use according to the 128 MB memory available on each node of the target computer.

# procs	4		8		16		32		64		128	
init.	2.58	2.58	2.57	2.57	2.57	2.57	2.57	2.57	2.57	2.57	2.57	2.57
setup local	0.99	0.80	1.00	1.01	1.40	1.31	1.40	1.30	1.40	1.31	1.40	1.32
setup coarse	0.25	0.00	0.48	0.00	0.86	0.00	0.85	0.00	0.87	0.00	0.93	0.00
iter.	1.84	1.98	2.55	3.93	3.01	5.14	4.12	7.16	3.79	9.80	4.91	13.26
total	5.33	5.65	6.23	7.26	7.14	8.50	8.25	10.51	7.93	13.13	9.14	16.60
# iter.	16	20	22	33	26	41	35	58	32	80	40	109
time per iter.	0.12	0.12	0.12	0.12	0.12	0.13	0.12	0.12	0.12	0.12	0.12	0.12

TABLE III.2.3: Elapsed time in each main numerical step varying the number of processors with 100×100 points per subdomain using the domain based coarse space.

# procs	4		8		16		32		64		128	
init.	2.58	2.58	2.57	2.57	2.57	2.57	2.57	2.57	2.57	2.57	2.57	2.57
setup local	0.66	0.80	0.94	1.01	1.24	1.31	1.24	1.30	1.24	1.31	1.25	1.32
setup coarse	0.72	0.00	0.74	0.00	0.90	0.00	0.90	0.00	0.92	0.00	1.07	0.00
iter.	1.85	2.31	1.97	3.93	2.10	5.14	2.36	7.16	2.03	9.80	2.45	13.26
total	5.61	5.65	6.09	7.26	6.65	8.50	6.91	10.51	6.59	13.13	7.16	16.60
# iter.	16	20	17	33	18	41	20	58	17	80	20	109
time per iter.	0.12	0.12	0.12	0.12	0.12	0.13	0.12	0.12	0.12	0.12	0.12	0.12

TABLE III.2.4: Elapsed time in each main numerical step varying the number of processors with 100×100 points per subdomain using the vertex based coarse space.

We can first observe that the numerical behavior of those preconditioners is again independent of the number of subdomains. It can be seen that the parallel implementation of the Schur complement method with only a local preconditioner scales perfectly as the time per iterations is constant and does not depend on the number of processors (i.e. 0.115 seconds on 4 processors and 0.118 on 128 nodes both figures were rounded to 0.12 seconds when reported in Table III.2.3 and III.2.4).

The above scalable behavior is also observed when the coarse components, vertex or subdomain, are introduced. For instance, with the vertex-based preconditioner, the time per iteration grows from 0.116 seconds on 4 processors up to 0.122 seconds on 128 processors (again rounded to 0.12 seconds in Table III.2.4). There are two main reasons for this scalable behavior. First, the solution of the coarse problems is negligible compared to the solution of the local Dirichlet problems. Second, the parallel implementation of the coarse components does not introduce any extra global communication.

In any case the methods scale fairly well, when the number of processors grows from 8 (to solve a problem with 80 000 unknowns) up to 128 (to solve a problem with 1.28 million unknowns). The ratios between the total elapsed time expended for running on 128 and on 8 processors are 1.18, with the vertex-based coarse preconditioner, and 1.47, with the domain-based one. That latter larger value is only due to an increase of the number of iterations.

One of the most expensive kernels of this method is the factorization of the local Dirichlet problems. Therefore, the tremendous reduction in the number of iterations induced by the use of the coarse alternatives - five times less iterations for the vertex-based preconditioner - is not reflected directly on a reduction of the total time. The total time is an affine function of the number of iterations with an incompressible overhead due to the Cholesky factorization at the very beginning of the domain decomposition method.

2.4 Concluding remarks

We have presented two-level preconditioners for Schur complement domain decomposition methods in two dimensions built by combining a variant of the of the local component of the BPS preconditioner with a set of new algebraic coarse space components.

Those numerical methods are targeted for parallel distributed memory computers. In this respect, we have proposed a message passing implementation that does not require any new global synchronization in the preconditioned conjugate gradient iterations, which is a well-known bottleneck for Krylov methods in distributed memory environments. We illustrated that the numerical scalability of the preconditioners combined with the parallel scalability of the implementation result in a set of parallel scalable numerical methods.

Chapter 3

Preliminary performance of overlapping domain decomposition in computational fluid dynamics

3.1 Introduction

The motivations to move from sequential to parallel computing are mainly twofold, that are either computing faster or computing larger. Depending on the objectives, the criterion to evaluate the quality of the parallel implementation is either the classical speed-up or the scaled speed-up [99]. The speed-up is usually defined by $\frac{T_1}{T_p}$, where T_i denotes the elapsed time on i processors to solve the target problem. It measures the gain introduced by the parallelization for solving a fixed size problem when the number of processors is varied. Using this definition, the speed-up is theoretically bounded by p when p processors are used. However super-linear speed-ups may be observed since increasing the number of processors usually implies decreasing the amount of data handled by each processor. This smaller amount of data better fit into the memory hierarchy of the processors resulting in faster computation. To overcome this drawback, the scaled speed-up can be considered. It is defined by $\frac{pT_1^*}{T_p^*}$ where T_p^* is the time required to solve a problem which size is proportional to the number of processors p. This latter criterion is usually preferred to evaluate the gain when parallel computing is considered to solve problems whose size is bounded by the memory available on the target parallel platform. The ideal situation is to have $T_1^* \approx T_p^*$, that is the case when a numerically scalable algorithm can be efficiently implement on a parallel computer platform. It can be mentioned that some parallel numerical algorithms are perfectly scalable in term of speed-up and poorly scalable in term of scaled speed-up. A class of such algorithms that exhibit this property are the explicit schemes for solving time dependent problems. They are straightforward to parallelize efficiently and linear speed-ups can be observed. Unfortunately in order to ensure the convergence stability of the numerical scheme the discretization time step should be reduced when the problem size is increased to evaluate the scaled speed-ups. This implies to perform more time steps to compute the solution at a given time and then leads to poor scaled speed-ups [84]. This fact mainly highlight the non numerical scalability of the explicit schemes when the mesh is refined.

In this chapter we mainly address the first situation that is computing faster the solution of a fixed size problem. As already mentioned, this study was developed in the framework on an industrial collaboration. The primary objective of this work was to reduce the elapsed time

required to compute the steady state solution of the Navier-Stokes equations on a set of large, but not huge problems, that might be tackled on a moderate number of processors. Therefore we only consider the classical speed-up to evaluate the efficiency of the parallel implementation. In addition, since this study was developed in a limited time period, the experimental results reported here should be considered as preliminary results and further investigations would deserve to be performed.

3.2 Parallel performance

In the sequel we report the speed-ups observed on the a IBM-SP2 using MPI as message passing library where we always assign one processor per subdomain. The test problem is the ONERA M6 wing, with a mesh composed by 77000 vertices and decomposed into 4, 8, 16 or 32 subdomains. The restart for GMRES was equal to 20. The tolerance for the stopping criterion is defined by the ratio of the 2-norm of the residual divided by the 2-norm of the right hand side and is set to 10^{-1}. All the experiments were performed in double precision arithmetic. Finally, because the problem was too large to fit into the memory of a single node of the SP2 we use the elapsed time on four processors to define the speed-up. That is the speed-up on p processors is evaluated by $\frac{T_4}{T_p}$.

The implementation of the additive Schwarz variants as preconditioner induced a significant reduction of the number of iterations but makes the iteration more costly both in term of memory space and CPU time. The preconditioner should be built prior the beginning of the GMRES iterations and applied at each step of the construction of the Krylov basis. If we consider the 4 processor example which history convergence is depicted in Figure II.3.2, $M_{ILU(0)-dAS}$ enables to converge in about five times less iterations than the original code that does not implement any preconditioner. However the overall computational time is only reduced by a factor of about four, decreasing from 7500 time units down to 2000.

# subdomains	4	8	16
Speed-up	1	1.86	3.44

TABLE III.3.1: speed-up using $M_{ILU(0)-dAS}$ - CFL=50.

# subdomains	4	8	16	32
Speed-up	1	1.99	3.97	7.02

TABLE III.3.2: speed-up using $M_{ILU(0)-dRAS}$ - CFL=50.

Table III.3.1 displays the speed-ups observed using the $M_{ILU(0)-dAS}$ variant as preconditioner. The parallel performance of $M_{ILU(0)-dRAS}$ are depicted in Table III.3.2. For both preconditioners, the good scalability of the speed-ups is partially due to a better memory/cache access resulting from the reduction of the local problem size handled by the processors. For $M_{ILU(0)-dAS}$, this memory effect partly alleviates the extra cost due to the slight increase of the number of iterations when the number of processors is increased; as it can be seen in Figure II.3.3. In contrast with $M_{ILU(0)-dRAS}$, where the surprising decrease of the overall number of Krylov steps (see Figure II.3.5) combined with the better memory access leads to remarkable observed speed-ups on eight and sixteen processors and reasonable good speed-up on thirty two processors.

3.3 Concluding remarks

The results presented in this chapter show that in a relative short period of time and moderate man-month effort, domain decomposition techniques can be implemented in an industrial code. These preconditioners enable to reduce the elapsed time required to perform numerical simulations.

Part IV

Conclusions and future work

At the time of the writing, this document is a snapshot of ongoing works that are still continuing to evolve and be developed. We review now the research directions that naturally emerge from this work.

In the framework of Jean-Christophe Rioual PhD thesis at CERFACS, that takes place within a joint collaboration with INRIA, non-overlapping domain decomposition techniques are used to parallelize a 2D device modeling code based on unstructured mixed finite element meshes [101]. Two-level preconditioners, including some of those described in this document, are investigating to design a scalable parallel simulation code. This work intensively uses the MUMPS code [4] since preliminary results indicate that the only local preconditioner that is robust enough in that context is the subdomain based that requires the explicit computation of the local Schur complement matrices.

It becomes more and more common now that one iterative solver has to be embedded in an outer one: this is the case, for instance, for solving eigenproblems with inverse iterations or with a Krylov method with invert. The question then arises: *what is the best strategy for stopping the inner iterations to ensure the convergence of the outer iterations while minimizing the global computational cost ?* Recently in the late nineties, the astonishing behavior of embedded solvers involving a Krylov outer process has been emphasized [22, 23, 95, 96]. Surprisingly, it is observed that the first Krylov vectors need to be known with full accuracy, and this accuracy can be significantly *relaxed* as the convergence proceeds. In [95], inner-outer iterations for the Conjugate Gradient are studied and applications are suggested such as saddle point problems. The work done in [22] on Krylov methods with inexact matrix-vector products has been extended to the context of Schur complement domain decomposition techniques where the local Dirichlet problems are solved with conjugate gradient iterations. Some further investigations would deserve to be performed to better understand this behavior and make the preliminary results reported in [24] applicable in a simulation code.

The numerical scalability of the CFD solver considered in this document should benefit from the use of an additional coarse space component in the preconditioner. Another alternative could be to use a non-overlapping Schur complement technique with inexact local solvers to design a scalable preconditioner. The idea is similar to the one presented in [143] but tuned to the finite element context. Some encouraging preliminary experiments have been performed on a scalar equation using this inexact Schur complement preconditioner for FGMRES, but its adequation in a CFD context needs to be assessed.

For the electromagnetism application, we intend to study the numerical scalability of the Frobenius norm minimization preconditioner when the size of the scaterred object is increased or when the wavelength of the illuminating wave is decreased resulting in larger linear systems. In this respect, we are implementing this preconditioner within a computational electromagnetism code that uses a fast multipole technique to perform the matrix-vector product. This work is continuing to be developed within the PhD thesis of Bruno Carpentieri at CERFACS. The combination of the fast multipole technique and the Frobenius norm minimization preconditioner should result in an efficient parallel code for solving huge problems.

The next improvement stems from addressing an emerging concern in computational electromagnetism. The objective is to solve linear systems with multiple right-hand sides using iterative schemes. Each right-hand side corresponds to illuminating an object with various waves that have the same wavelength but correspond to different angles of incidence. For this purpose, block Krylov techniques will be studied. This work is being developed in the context of a joint collaboration with Aerospatiale through the PhD thesis of Julien Langou at CERFACS. One of the numerical difficulties in this context is to be able to detect and manage the situation where one right-hand side or a linear combination of right-hand sides have converged before the others.

Those subjects will certainly constitute most of my near future research activities that will

probably be developed on the emerging parallel platforms that are the clusters of symmetric multiprocessors (SMP). On these parallel platforms it is not clear whether one should mix the two parallel paradigms, that are shared memory programming, through OpenMP directives, and message passing, via MPI, or if a complete distribute memory approach will still enable to fully exploit the capabilities of those architectures for the class of algorithms discussed in this document.

Bibliography

[1] R. Aitbayev, X.-C. Cai, and M. Paraschivoiu. Parallel two-level methods for three-dimensional transonic compressible flow simulations on unstructured meshes. In *Parallel CFD'99*, 1999.

[2] G. Alléon, M. Benzi, and L. Giraud. Sparse approximate inverse preconditioning for dense linear systems arising in computational electromagnetics. *Numerical Algorithms*, 16:1–15, 1997.

[3] Ch. Amestoy, M. J. Daydé, and L. Giraud. Parallélisation conjointe CERFACS-CNES d'un code pilote du CNES. Final Contract Report FR/PA/94/13, CERFACS, Toulouse, France, 1994.

[4] P. R. Amestoy, I. S. Duff, and J. Y. L'Excellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. *to appear in special issue of Comput. Methods in Appl. Mech. Eng. on domain decomposition and parallel computing*, 1998.

[5] D. Amitai, A. Averbuch, M. Israeli, and S. Itzikowitz. Implicit-explicit parallel asynchronous solver for PDEs. *SIAM J. Scientific Computing*, 19:1366–1404, 1998.

[6] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Oustrouchov, and D. Sorensen. *LAPACK - Users' Guide.* SIAM, Philadelphia, 2nd edition, 1995.

[7] O. Axelsson. *Iterative Solution Methods.* Cambridge University Press, Cambridge, 1994.

[8] S. Baldini, L. Giraud, L. Hamel, J. M. Jimenez, and L. M. Matey. HIPERCOMBATS: a parallel industrial tool for two-wheeler suspensions design. In B. Hertzberger and P. Sloot, editors, *High Performance Computing and Networking*, pages 51–59, 1997.

[9] S. Baldini, L. Giraud, J. M. Jimenez, L. M. Matey, and J. G. Izaguirre. High performance computing in multi-body system design. *Int J. Supercomputer Applications*, 13(2):99–106, 1999.

[10] B. Barán, E. Kaszkurewicz, and A. Bhaya. Parallel asynchronous team algorithms: convergence and performance analysis. *IEEEE Transactions on Parallel and Distributed Systems*, 7:677–688, 1996.

[11] A. Beguelin, J. Dongarra, A. Geist, W. Jiang, R. Manchek, and V. Sunderam. PVM 3 user's guide and Reference Manual. Technical Report ORNL/TM-12187, Oak Ridge National Laboratory, Tennessee 37831, 1994.

[12] A. Bendali. *Approximation par éléments finis de surface de problèmes de diffraction des ondes électro-magnétiques.* PhD thesis, *Université Paris VI* , 1984.

[13] M. W. Benson. Iterative solution of large scale linear systems. Master's thesis, Lakehead University, Thunder Bay, Canada, 1973.

[14] M. W. Benson and P. O. Frederickson. Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems. *Utilitas Mathematica*, 22:127–140, 1982.

[15] M. W. Benson, J. Krettmann, and M. Wright. Parallel algorithms for the solution of certain large sparse linear systems. *Int J. of Computer Mathematics*, 16:245–260, 1984.

[16] M. Benzi and L. Giraud. Acquisition d'une méthode pour la résolution des systèmes linéaires issus des problèmes d'électromagnétisme. Final Contract Report FR/PA/97/04, CERFACS, Toulouse, France, 1997.

[17] M. Benzi, C. D. Meyer, and M. Tůma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM J. Sci. Comput.*, 17(5):1135–1149, 1996.

[18] M. Benzi and M. Tůma. Approximate inverse preconditioning of iterative methods for non-symmetric linear systems. *SIAM*, 19(3):968–994, 1998.

[19] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and distributed computation.* Printice Hall, Englewood Cliffs, New Jersey, 1989.

[20] D. P. Bertsekas and J. N. Tsitsiklis. Some aspects of parallel and distributed iterative algorithms - A survey. *Automatica*, 27:3–21, 1991.

[21] P. E. Bjørstad, M. S. Espedal, and D. E. Keyes, editors. *Proc. Ninth Int. Conf. on Domain Decomposition Meths.* Domain decomposition press, Bergen, 1998.

[22] A. Bouras and V. Frayssé. A relaxing strategy for inexact matrix-vector products for Krylov methods. Technical Report TR/PA/00/15, CERFACS, Toulouse, France, 2000.

[23] A. Bouras and V. Frayssé. A relaxing strategy for the arnoldi method in eigenproblems. Technical Report TR/PA/00/16, CERFACS, Toulouse, France, 2000.

[24] A. Bouras, V. Frayssé, and L. Giraud. A relaxation strategy for inner-outer linear solvers in domain decomposition methods. Technical Report TR/PA/00/17, CERFACS, Toulouse, France, 2000.

[25] J. F. Bourgat, R. Glowinski, P. Le Tallec, and M. Vidrascu. Variational formulation and algorithm for trace operator in domain decomposition calculations. In T. Chan, R. Glowinski, J. Périaux, and O. Widlund, editors, *Second International Conference on Domain Decomposition Methods.* SIAM, Philadelphia, PA, 1989.

[26] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring I. *Math. Comp.*, 47(175):103–134, 1986.

[27] P. N. Brown and Y. Saad. Hybrid Krylov methods for nonlinear systems of equations. *SIAM J. Sci. Stat. Comput.*, 11:59–71, 1990.

[28] P. N. Brown and Y. Saad. Convergence theory of nonlinear Newton-Krylov algorithms. *SIAM J. Optimization*, 4:297–330, 1994.

[29] X.-C. Cai, C. Farhat, and M. Sarkis. A minimum overlap restricted additive Schwarz preconditioner and applications in 3D flow simulations. In *Proceedings of the 10th Intl. Conf. on Domain Decomposition Methods*, pages 479–485, Providence, 1998. AMS.

[30] X.-C. Cai, W. D. Gropp, D. E. Keyes, R. G. Melvin, and D. P. Young. Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation. *SIAM J. Sci. Stat. Comput.*, 19:246–265, 1998.

[31] X.-C. Cai, D. E. Keyes, and V. Venkatakrishnan. Newton-Krylov-Schwarz: an implicit solver for CFD. In R. Glowinski, J. Périaux, Z.-C. Shi, and O. B. Widlund, editors, *Eighth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, Chichester, 1996. Wiley and Sons.

[32] X.-C. Cai and M. Sarkis. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM J. Sci. Stat. Comput.*, 21:513–521, 1999.

[33] B. Carpentieri, I. S. Duff, and L. Giraud. Sparse pattern selection strategies for robust Frobenius norm minimization preconditioners in electromagnetism. *Numerical Linear Algebra with Applications*, to appear, 2000.

[34] B. Carpentieri, I.S. Duff, and L.Giraud. Experiments with sparse preconditioning of dense problems from electromagnetic applications. Technical Report TR/PA/00/04, CERFACS, Toulouse, France, 2000.

[35] L. M. Carvalho. *Preconditioned Schur complement methods in distributed memory environments.* PhD thesis, INPT/CERFACS, Toulouse, France, october 1997. TH/PA/97/41, CERFACS.

[36] L. M. Carvalho and L. Giraud. Additive Schwarz for the Schur complement method. In *Domain Decomposition Methods in Scientific Computing*, pages 304–310. Domain Decomposition Press, Bergen, 1998.

[37] L. M. Carvalho and L. Giraud. Block diagonal preconditioners for the Schur complement method. In M. Papadrakakis and B.H.V. Topping, editors, *Innovative computational methods for structural mechanics*, pages 55–76, Edinburgh, UK, 1999. Saxe-Coburg publications.

[38] L.M. Carvalho, L. Giraud, and G. Meurant. Local preconditioners for two-level non-overlapping domain decomposition methods. Tech. Rep. TR/PA/99/38, CERFACS, France, 1999. Preliminary version of the paper to appear in Numerical Linear Algebra with Applications.

[39] L.M. Carvalho, L. Giraud, and P. Le Tallec. Algebraic two-level preconditioners for the Schur complement method. Tech. Rep. TR/PA/98/18, CERFACS, France, 1998. Preliminary version of the paper to appear in SIAM SISC.

[40] F. Chalot, G. Chevalier, Q.V. Dinh, and L. Giraud. Some investigations of domain decomposition techniques in parallel CFD. In P. Amestoy, Ph. Berger, M. Daydé, V. Frayssé, L. Giraud, and D. Ruiz, editors, *EUROPAR'99 Parallel Processing*, volume 1685, pages 595 – 602. Lecture Notes in Computer Science, Springer-Verlag, 1999.

[41] T. F. Chan, R. Glowinski, J. Périaux, and O. B. Widlund, editors. *Proc. Second Int. Conf. on Domain Decomposition Meths.* SIAM, Philadelphia, 1989.

[42] T. F. Chan, R. Glowinski, J. Périaux, and O. B. Widlund, editors. *Proc. Third Int. Conf. on Domain Decomposition Meths.* SIAM, Philadelphia, 1990.

[43] T. F. Chan and T. P. Mathew. The interface probing technique in domain decomposition. *SIAM J. Matrix Analysis and Applications*, 13:212–238, 1992.

[44] T. F. Chan and T. P. Mathew. *Domain Decomposition Algorithms*, volume 3 of *Acta Numerica*, pages 61–143. Cambridge University Press, Cambridge, 1994.

[45] T. F. Chan, T. P. Mathew, and J-P. Shao. Fourier and probe variants of the vertex space domain decomposition algorithm. In D. Keyes, T. F. Chan, G. Meurant, J. Scroggs, and R. Voigt, editors, *Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 236–249, Phil., 1992. SIAM.

[46] D. Chazan and W. Miranker. Chaotic relaxation. *Linear Algebra and its Applications*, 2:199–222, 1969.

[47] K. Chen. On a class of preconditioning methods for dense linear systems from boundary elements. *SIAM J. Scientific Computing*, 20(2):684–698, 1998.

[48] E. Chow. A priori sparsity patterns for parallel sparse approximate inverse preconditioners. Tech. Rep UCRL-JC-130719, Lawrence Livermore National Laboratory, Livermore, CA, 1998. To appear in SISC.

[49] E. Chow and Y. Saad. Approximate inverse preconditioners via sparse-sparse iterations. *SIAM J. Scientific Computing*, 19(3):995–1023, 1998.

[50] F. Collino, S. Ghanemi, L. Giraud, S. Gratton, M. Invernizzi, P. Joly, and A. Piacentini. Rapport de fin de contrat Thomson: Résolution des équations de Maxwell tridimensionnelles dans le domaine fréquentiel sur réseaux hétérogènes de calculateurs. Final Contract Report FR/EL-PA/95/25, CERFACS, Toulouse, France, 1995.

[51] M. J. Daydé, I. S. Duff, J. Y. L'Excellent, and L. Giraud. Evaluation d'ordinateurs vectoriels et parallèles sur un jeu de programmes représentatifs des calculs intensifs à la division avions de l'Aerospatiale. Final Contract Report PR/PA/93/19, CERFACS, Toulouse, France, 1993.

[52] Y.-H. De Roeck and P. Le Tallec. Analysis and test of a local domain decomposition preconditioner. In R. Glowinski, Y. Kuznetsov, G. Meurant, J. Périaux, and O. Widlund, editors, *Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 112–128. SIAM, Philadelphia, PA, 1991.

[53] E. de Sturler and H. A. van der Vorst. Communication cost reduction for krylov methods on parallel computers. In *Lecture Notes in Computer Science, High-Performance Computing and Networking*, volume 796, pages 190–195. Springer-Verlag, April 1994.

[54] G. Degrez, L. Giraud, M. Loriot, A. Micelotta, B. Nitrosso, and A. Stoessel. Parallel industrial CFD calculations with N3S. In *Lecture Notes in Computer Science, High-Performance Computing and Networking*, volume 919, pages 820–825. Springer-Verlag, May 1995.

[55] B. Dembart and M. A. Epton. Low frequency multipole translation theory for the Helmholtz equation. Tech. Rep SSGTECH-98-013, The Boeing Company, Seattle, WA, 1998.

[56] B. Dembart and M. A. Epton. Spherical harmonic analysis and synthesis for the fast multipole method. Tech. Rep SSGTECH-98-014, The Boeing Company, Seattle, WA, 1998.

[57] B. Dembart and E. Yip. Matrix assembly in FMM-MOM codes. Tech. Rep ISSTECH-97-002, The Boeing Company, Seattle, WA, 1997.

[58] S. Demko, W.F. Moss, and P.W. Smith. Decay rates for inverses of band matrices. *Mathematics of Computation*, 43:491–499, 1984.

[59] R. D. Dowsing. *Introduction to concurrency using OCCAM.* Chapman & Hall, 1998.

[60] M. Dryja, B. F. Smith, and O. B. Widlund. Schwarz analysis of iterative substructuring algorithms for elliptic problems in three dimensions. *SIAM J. Numer. Anal.*, 31:1662–1694, 1994.

[61] M. Dryja and O. B. Widlund. Towards a unified theory of domain decomposition algorithms for elliptic problems. In T. F. Chan, R. Glowinski, J. Périaux, and O. Widlund, editors, *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 3–21. SIAM, Philadelphia, PA, 1990.

[62] I.S. Duff and J.K. Reid. MA27 – A set of Fortran subroutines for solving sparse symmetric sets of linear equations. Technical Report AERE R10533, Rutherford Appleton Laboratory, London, 1982.

[63] D. El Baz. A method of terminating asynchronous iterative algorithms on message passing systems. *Parallel Algorithms and Applications*, (9):153–158, 1996.

[64] D. El Baz, P. Spitéri, and J. C. Miellou. Distributed asynchronous iterative methods with order intervals for a class of nonlinear optimization problems. *Journal of Parallel and Distributed Computing*, 38:1–15, 1996.

[65] M. N. El Tarazi. Some convergence results for asynchronous algorithms. *Numerische Mathematik*, 39:325–340, 1984.

[66] C. Farhat and F.-X. Roux. A method of finite element tearing and interconnecting and its parallel solution algorithm. *Int. J. Numer. Meth. Engng.*, 32:1205–1227, 1991.

[67] V. Frayssé and L. Giraud. An implementation of block QMR for J-symmetric matrices. Final Contract Report FR/PA/97/57, CERFACS, Toulouse, France, 1997.

[68] V. Frayssé, L. Giraud, and S. Gratton. Solveurs linéaires itératifs pour la résolution de systèmes complexes non hermitiens creux de grande taille. Final Contract Report FR/PA/96/34, CERFACS, Toulouse, France, 1996.

[69] V. Frayssé, L. Giraud, and S. Gratton. A set of GMRES routines for real and complex arithmetics. Tech. Rep. TR/PA/97/49, CERFACS, 1997.

[70] V. Frayssé, L. Giraud, and S. Gratton. A set of flexible-GMRES routines for real and complex arithmetics. Tech. Rep. TR/PA/98/20, CERFACS, France, 1998.

[71] V. Frayssé, L. Giraud, and H. Kharraz-Aroussi. On the influence of the orthogonalization scheme on the parallel performance of GMRES. In D. Pritchard and J. Reeve, editors, *EUROPAR'98 Parallel Processing*, volume 1470, pages 751–762. Springer, 1998.

[72] P. O. Frederickson. Fast approximate inversion of large sparse linear systems. Math. Report 7, Lakehead University, Thunder Bay, Canada, 1975.

[73] R. W. Freund. A transpose-free quasi-minimal residual algorithm for non-hermitian linear systems. *SIAM J. Scientific Computing*, 14(2):470–482, 1993.

[74] R. W. Freund and M. Malhotra. A block QMR algorithm for non-hermitian linear systems with multiple right-hand sides. *Linear Algebra and Its Applications*, 254:119–157, 1997.

[75] R. W. Freund and N. M. Nachtigal. QMR: a quasi-minimal residual method for non-hermitian linear systems. *Numerische Mathematik*, 60(3):315–339, 1991.

[76] R. W. Freund and N. M. Nachtigal. QMRPACK: a package of QMR algorithms. *ACM Transactions on Mathematical Software*, 22:46–77, 1996.

[77] A. Frommer, H. Schwandt, and D. B. Szyld. Asynchronous weighted additive schwarz methods. *Electronic Transactions on Numerical Analysis*, 5:48–61, 1997.

[78] A. Frommer and D. B. Szyld. On asynchronous iterations. Technical Report Report 99-5-31, Temple University, 1999. To appear in Journal of computational Applied Mathematics.

[79] A. George and J. W. Liu. *Computer solution of large sparse positive definite systems.* Prentice-Hall series in Computational Mathematics, Englewood Cliffs, 1981.

[80] L. Giraud. *Implantations parallèles de méthodes de sous-domaines synchrones et asynchrones pour la résolution de problèmes aux limites.* PhD thesis, Institut National Polytechnique de Toulouse, 1991.

[81] L. Giraud, R. Guivarch, and J. Stein. A parallel distributed fast 3D Poisson solver for MésoNH. In P. Amestoy, Ph. Berger, M. Daydé, V. Frayssé, L. Giraud, and D. Ruiz, editors, *EUROPAR'99 Parallel Processing*, volume 1685, pages 1431 – 1434. Lecture Notes in Computer Science, Springer-Verlag, 1999.

[82] L. Giraud, M. Henry, L. Lefebvre, L. Perret, C. Puglisi, and D. Vaucher. Parallelization of ASTRYD, a software based on a time domain method for solving problems in vibroacoustics. volume 2, pages 533–538. CETIM, 1995.

[83] L. Giraud, N. Maman, P. Menegazzi, A. Micelotta, and B. Thomas. Parallel industrial incompressible cfd calculations with HPCN3S. In H. Liddell, A. Colbrook, B. Hertzberger, and P. Sloot, editors, *High-Performance Computing and Networking*, volume 1067, pages 122–127. Springer-Verlag, 1996.

[84] L. Giraud and G. Manzini. Parallel implementations of 2D explicit Euler solvers. *Journal of computational physics*, 123:111–118, 1996.

[85] L. Giraud and P. Spitéri. Résolution parallèle de problèmes aux limites non-linéaires. *Modélisation Mathématique et Analyse Numérique*, 25(4):579–606, 1991.

[86] L. Giraud and R. Tuminaro. A domain decomposition probing variant suitable for anisotropic problems. Technical Report TR/PA/93/36, CERFACS, Toulouse, France, 1993.

[87] L. Giraud and R. Tuminaro. Grid transfer operators for highly variable coefficient problems. Technical Report TR/PA/93/37, CERFACS, Toulouse, France, 1993.

[88] L. Giraud and R. S. Tuminaro. Domain decomposition algorithms for the drift-diffusion equations. In R. Sincovec, D. Keyes, M. Leuze, L. Petzold, and D. Reed, editors, *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing*, pages 719–726. SIAM, 1993.

[89] L. Giraud and R. S. Tuminaro. Schur complement preconditioners for anisotropic problems. *IMA J. Numerical Analysis*, 19(1):1–17, 1999.

[90] R. Glowinski, G. H. Golub, G. A. Meurant, and J. Périaux, editors. *Proc. First Int. Conf. on Domain Decomposition Meths.* SIAM, Philadelphia, 1988.

[91] R. Glowinski, Yu. A. Kuznetsov, G. A. Meurant, J. Périaux, and O. B. Widlund, editors. *Proc. Fourth Int. Conf. on Domain Decomposition Meths.* SIAM, Philadelphia, 1991.

[92] R. Glowinski, J. Périaux, Z.-C. Shi, and O. B. Widlund, editors. *Proc. Eighth Int. Conf. on Domain Decomposition Meths.* Wiley and Sons, Chichester, 1996.

[93] G. H. Golub and D. Mayers. The use of preconditioning over irregular regions. In R. Glowinski and J.L. Lions, editors, *Computing Methods in Applied Sciences and Engineering, VI*, pages 3–14, Amsterdam, 1984. North-Holland.

[94] G. H. Golub and C. F. van Loan. *Matrix Computations.* Johns Hopkins Univ. Press, Baltimore, MD, 1996. Third edition.

[95] G. H. Golub and Q. Ye. Inexact preconditioned conjugate gradient method with inner-outer iteration. Sccm-97-04, Stanford University, 1997.

[96] G. H. Golub, Z. Zhang, and H. Zha. Large sparse symmetric eigenvalue problems with homogeneous linear constraints: the Lanczos process with inner-outer iterations. *Lin. Alg. Appl.*, 2000. to appear.

[97] N.I.M. Gould and J.A. Scott. Sparse approximate-inverse preconditioners using norm-minimization techniques. *SIAM J. Scientific Computing*, 19(2):605–625, 1998.

[98] M. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM J. Sci. Comput.*, 18:838–853, 1997.

[99] J. Gustafson, G. Montry, and R. Benner. Development of parallel methods for a 1024-processor hypercube. *SIAM J. Sci. Stat. Comput.*, 9(4):609–638, July 1988.

[100] W. Hackbusch. *Multi-grid Methods and Applications.* Springer-Verlag, Berlin, 1985.

[101] F. Hecht and A. Marrocco. Mixed finite element simulation of heterojonction structures including a boundary layer model for the quasi-fermi levels. *COMPEL*, 13:757–770, 1995.

[102] B. Hendrikson and R. Leland. The Chaco user's guide: Version 2.0. Technical Report SAND94-2692, Sandia National Laboratories, 1994.

[103] C. A. R. Hoare. *Processus séquentiels communiquants.* Masson, Paris, France, 1987.

[104] R. .A. Howard. *Dynamic programming and Markov process.* MIT Press, 1960.

[105] HSL. A collection of Fortran codes for large scale scientific computation, 2000. http://www.numerical.rl.ac.uk/hsl.

[106] T. J.R. Hughes, L. P. Franca, and M. Mallet. A new finite element formulation for computational fluid dynamics: VI. convergence analysis of the generalized SUPG formulation for linear time-dependant multidimentional advective-diffusive systems. *Comp. Meth. in Applied Mechanics and Engineering*, (63):97–112, 1987.

[107] T. J.R. Hughes and M. Mallet. A new finite element formulation for computational fluid dynamics: IV. a discontinuity-capturing operator for multidimentional advective diffusive systems. *Comp. Meth. in Applied Mechanics and Engineering*, (58):329–339, 1986.

[108] P. Jabouille, R. Guivarch, P. Kloos, D. Gazen, N. Gicquel, L. Giraud, N. Asencio, V. Ducrocq, J. Escobar, J.-L. Redelsperger, J. Stein, and J.-P. Pinty. Parallelization of the french meteorological mesoscale model MésoNH. In P. Amestoy, Ph. Berger, M. Daydé, V. Frayssé, L. Giraud, and D. Ruiz, editors, *EUROPAR'99 Parallel Processing*, volume 1685, pages 1417 – 1422. Lecture Notes in Computer Science, Springer-Verlag, 1999.

[109] J. M. Jimenez, L. Matey, J. Garcia, A. Avello, S. Baldini, L. Giraud, and L. Hamel. On the use of high-performance computing in multi-body analysis for two-wheeled suspension design. In H. Rahnejat and R. Whalley, editors, *Multi-Body Dynamics: Monitoring and Simulation Techniques*, pages 295–304. Mechanical Engineering Publications Limited, 1997.

[110] G. Karypis and V. Kumar. METIS - A software for partitioning unstructured graphs, partitionong meshes and computing fill-reduction orderings of sparse matrices - Version 4.0. Technical Report , Universityy of Minnesota, 1998.

[111] D. E. Keyes, T. F. Chan, G. A. Meurant, J. S. Scroggs, and R. G. Voigt, editors. *Proc. Fifth Int. Conf. on Domain Decomposition Meths.* SIAM, Philadelphia, 1992.

[112] D. E. Keyes and J. Xu, editors. *Proc. Seventh Int. Conf. on Domain Decomposition Meths.* Number 180 in Contemporary Mathematics. AMS, Providence, 1995.

[113] L.Yu. Kolotilina. Explicit preconditioning of systems of linear algebraic equations with dense matrices. *J. Sov. Math*, 43:2566–2573, 1988. English translation of a paper first published in Zapisli Nauchnykh Seminarov Leningradskogo Otdeleniya Matematicheskogo im. V.A. Steklova AN SSSR 154 (1986) 90-100.

[114] L.Yu Kolotilina and A.Yu. Yeremin. Factorized sparse approximate inverse preconditionings I: Theory. *SIAM J. Matrix Analysis and Applications*, 14:45–58, 1993.

[115] C.-H. Lai, P. E. Bjørstad, M. Cross, and O. Widlund, editors. *Proc. Eleventh Int. Conf. on Domain Decomposition Meths.* Domain decomposition press, Bergen, 1999.

[116] P. Le Tallec. *Domain decomposition methods in computational mechanics*, volume 1 of *Computational Mechanics Advances*, pages 121–220. North-Holland, 1994.

[117] J.-L. Lions. On the schwarz alternating method. I. In R. Glowinski, G. Golub, and J. Periaux, editors, *Proceedings of the first international symposium on domain decomposition methods for partial differential equations*, Paris, France, January 7-9, 1987 1988. SIAM, Philadelphia.

[118] J. Mandel. Balancing domain decomposition. *Communications in Numerical Methods in Engineering*, 9:233–241, 1993.

[119] J. Mandel and M. Brezina. Balancing domain decomposition: Theory and computations in two and three dimensions. Technical Report UCD/CCM 2, Center for Computational Mathematics, University of Colorado at Denver, 1993.

[120] J. Mandel and M. Brezina. Balancing domain decomposition for problems with large jumps in coefficients. *Mathematics of Computation*, 65:1387–1401, 1996.

[121] J. Mandel, C. Farhat, and X.-C. Cai, editors. *Proc. Tenth Int. Conf. on Domain Decomposition Meths.* Number 218 in Contemporary Mathematics. AMS, Providence, 1998.

[122] J. Mandel and R. Tezaur. Convergence of substructuring method with Lagrange multipliers. *Numer. Math.*, 73:473–487, 1996.

[123] J. A. Meijerink and H. A. van der Vorst. An iterative solution method for linear systems for which the coefficient matrix is a symmetric M-matrix. *Mathematics od Computations*, 31(137):148–162, 1977.

[124] Message Passing Interface Forum. MPI: A message-passing interface standard. Int. J. Supercomputer Applications and High Performance Computing, 8(3/4), 1994. Special issue on MPI.

[125] G. A. Meurant. Multitasking the conjugate gradient method on the CRAY X-MP/48. *Parallel Comput.*, 7:267–280, 1987.

[126] G. A. Meurant. Numerical experiments with a domain decomposition method for parabolic problems. In R. Glowinski, Yu. A. Kuznetsov, G. A. Meurant, J. Périaux, and O. B. Widlund, editors, *Proc. Fourth Int. Conf. on Domain Decomposition Meths.*, pages 394–408, Philadelphia, 1991. SIAM.

[127] G. A. Meurant. *Computer solution of large linear systems*, volume 28 of *Studies in Mathematics and its Applications*. North-Holland, 1999.

[128] J. C. Miellou. Algorithmes de relaxation chaotiques à retard. *RAIRO R1*, pages 55–82, 1975.

[129] J. C. Miellou. Asynchronous iterations in order intervals. In M. Cosnard et al., editor, *Parallel Algorithms*, pages 85–96. Amsterdam: North-Holland, 1986.

[130] J. C. Miellou, D. El Baz, and P. Spitéri. A new class of asynchronous iterative algorithms with order intervals. *Mathematics od Computations*, 67:237–255, 1998.

[131] J. C. Miellou and P. Spitéri. Un critère de convergence pour des méthodes générales de point fixe. *Modélisation Mathématique et Analyse Numérique*, pages 645–669, 1985.

[132] L. Paglieri, A. Scheinine, L. Formaggia, and A. Quarteroni. Parallel conjugate gradient with Schwarz preconditioner applied to fluid dynamics problems. In P. Schiano et al., editor, *Parallel Computational Fluid Dynamics, Algorithms and Results using Advanced Computer, Proceedings of Parallel CFD'96*, pages 21–30, 1997.

[133] V. Pereyra. Asynchronous distributed solution of large scale nonlinear inversion problems. *Applied Numerical Mathematics*, 30:31–40, 1999.

[134] M. Pott. On the convergence of asynchronous iterations methods for nonlinear paracontractions and consistent linear systems. *Linear Algebra and its Applications*, 283:1–33, 1998.

[135] A. Quarteroni, J. Périaux, Yu. A. Kuznetsov, and O. B. Widlund, editors. *Proc. Sixth Int. Conf. on Domain Decomposition Meths.* Number 157 in Contemporary Mathematics. AMS, Providence, 1994.

[136] A. Quarteroni and A. Valli. *Domain decomposition methods for partial differential equations.* Numerical mathematics and scientific computation. Oxford science publications, Oxford, 1999.

[137] J. Rahola. Experiments on iterative methods and the fast multipole method in electromagnetic scattering calculations. Technical Report TR/PA/98/49, CERFACS, Toulouse, France, 1998.

[138] F. Robert. Contraction en norme vectorielle : convergence d'itération chaotqiue. *Linear Algebra and its Applications*, (13):19–35, 1975.

[139] F. Robert, M. Charnay, and F. Musy. Itérations chaotiques série parallèle pour des équations non-linéaires de point fixe. *Aplikace Mathematik*, (20):1–38, 1975.

[140] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput.*, 14:461–469, 1993.

[141] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing, New York, 1996.

[142] Y. Saad and M. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[143] Y. Saad and M. Sosonkina. Distributed Schur complement techniques for general sparse linear systems. *SIAM J. Scientific Computing*, 21(4):1337 – 1356, 2000.

[144] S. A. Savari and D. P. Bertserkas. Finite termination of asynchronous iterative algorithms. *Parallel computing*, 22:39–56, 1996.

[145] H.A. Schwarz. Über eine grenzübergang durch alternirendes verfahren. *Gesammelete Mathematische Abhandlungen, Springer-Verlag*, 2:133–143, 1890. First published in Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich, vol.15, pp. 272-286, 1870.

[146] B. F. Smith. *Domain Decomposition Algorithms for the Partial Differential Equations of Linear Elasticity.* PhD thesis, Courant Institute of Mathematical Sciences, September 1990. Tech. Rep. 517, Department of Computer Science, Courant Institute.

[147] B. F. Smith, P. Bjørstad, and W. Gropp. *Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Cambridge University Press, New York, 1st edition, 1996.

[148] P. Spitéri, R. Guivarch, H. Boisson, and J. C. Miellou. Schwarz alternating parallel algorithm applied to incompressible flows computation in vorticity stream function formulation. *Parallel Algorithms and Applications*, 11:205–225, 1998.

[149] X-C. Tai and P. Tseng. Convergence rate analysis of an asynchronous space decomposition method for convex minimization. Technical Report Technical report 120, Department of Mathematics, University of Bergen, Norway, 1998.

[150] P. Tseng. On the rate of convergence of a partially asynchronous gradient projection algorithm. *SIAM J. Control and Optimization*, 1:603–619, 1991.

[151] P. Tseng, D. P. Bertsekas, and J. N. Tsitsiklis. Partially asynchronous parallel algorithm for network flow and other problems. *SIAM J. Control and Optimization*, 28:678–710, 1990.

[152] 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. Scientific and Statistical*, 13:631–644, 1992.

[153] H. A. van der Vorst and C. Vuik. GMRESR: a family of nested GMRES methods. *Numerical Linear Algebra with Applications*, 1(4):369–386, 1994.

[154] W. L. Wan. *Scalable and multilevel iteratives methods.* PhD thesis, Department of Mathematics, UCLA, Los Angeles, June 1998.

[155] W. L. Wan, T. F. Chan, and B. Smith. An energy-minimizing interpolation for robust multigrid. *SIAM J. Scientific Computing*, 21(4):1632 – 1649, 2000.

[156] P. Wesseling. *An Introduction to Multigrid Methods.* Wiley, West Sussex, 1992.

[157] J. Zdeněk, T. J.R. Hughes, and S. Farzin. A globally convergent matrix-free algorithm for implicit time marching schemes arising in finite element analysis in fluids. *Comp. Meth. in Applied Mechanics and Engineering*, 1991.

Appendix: complete list of publications

Papers in journal

[1] G. Alléon, M. Benzi, and L. Giraud, (1997), Sparse Approximate Inverse Preconditioning for Dense Linear Systems Arising in Computational Electromagnetics, *Numerical Algorithms*, **16**, 1–15.

[2] S. Baldini, L. Giraud, J. M. Jimenez, L. M. Matey, and J. G. Izaguirre, (1999), High Performance Computing in Multi-Body System Design, *Int J. Supercomputer Applications*, **13**, 99–106.

[3] B. Carpentieri, I. S. Duff, and L. Giraud, Sparse pattern selection strategies for robust Frobenius norm minimization preconditioners in electromagnetism, *Numerical Linear Algebra with Applications*, **to appear**.

[4] L.M. Carvalho, L. Giraud, and G. Meurant. Local preconditioners for two-level non-overlapping domain decomposition methods. *Numerical Linear Algebra with Applications*, **to appear**.

[5] L.M. Carvalho, L. Giraud, and P. Le Tallec. Algebraic two-level preconditioners for the Schur complement method. *SIAM J. Scientific Computing*, **to appear**.

[6] L. Giraud and G. Manzini, (1996), Parallel implementations of 2D explicit Euler solvers, *Journal of computational physics*, **123**, 111–118.

[7] L. Giraud and P. Spitéri, (1991), Résolution parallèle de problèmes aux limites non-linéaires, *Modélisation Mathématique et Analyse Numérique*, **25**, 579–606.

[8] L. Giraud and R. S. Tuminaro, (1995), Time dependent solvers on distributed memory computers, *Calculateurs parallèles*, **7**, 255–269.

[9] L. Giraud and R. S. Tuminaro, (1999), Schur Complement Preconditioners for Anisotropic Problems, *IMA J. Numerical Analysis*, **19**, 1–17.

[10] L. Giraud, J. Miellou, and P. Spitéri, (1992), S.S.O.R. preconditioning behaviour with respect to the relaxation parameter, in case of by plane discretization of 3D-problems, *Intern. J. Computer Math.*, **40**, 153–158.

[11] L. Giraud, (1995), Block preconditioned conjugate gradient methods on a distributed virtual shared memory multiprocessor, *Int J. High Speed Computing*, **7**, 161–190.

Papers in proceedings

[1] S. Baldini, L. Giraud, L. Hamel, J. M. Jimenez, and L. M. Matey, (1997), HIPERCOMBATS: a Parallel Industrial Tool for Two-Wheeler Suspensions Design, In *High Performance Computing and Networking*, B. Hertzberger and P. Sloot, eds., 51–59.

[2] P. Berger, L. Giraud, and P. Spitéri, (1989), Parallel asynchronous 2D Poisson Equation solvers on a processor network, In *Computational Mechanics Publications*, vol. 2, Springer-Verlag, 265–271.

[3] L. M. Carvalho and L. Giraud, (1996), Parallel Domain Decomposition Experiments on the Meiko CS2-HA, In *High-Performance Computing and Networking*, H. Liddell, A. Colbrook, B. Hertzberger, and P. Sloot, eds., vol. 1067, Springer-Verlag, 948–949.

[4] L. M. Carvalho and L. Giraud, (1998), Additive Schwarz for the Schur complement method, In *Domain Decomposition Methods in Scientific Computing*, Domain Decomposition Press, Bergen, 304–310.

[5] L. M. Carvalho and L. Giraud, (1999a), Block diagonal preconditioners for the Schur complement method, In *Innovative computational methods for structural mechanics*, M. Papadrakakis and B. Topping, eds., Edinburgh, UK, Saxe-Coburg publications, 55–76.

[6] L. M. Carvalho and L. Giraud, (1999b), Parallel subdomain-based preconditioner for the Schur complement, In *EUROPAR'99 Parallel Processing*, P. Amestoy, P. Berger, M. Daydé, V. Frayssé, L. Giraud, and D. Ruiz, eds., vol. 1685, Lecture Notes in Computer Science, Springer-Verlag, 1032 –1039.

[7] L. M. Carvalho, I. S. Duff, and L. Giraud, (1996), Linear algebra kernels for parallel domain decomposition methods, In *Advanced Computational Methods in Structural Mechanics*, M. Papadrakakis and G. Bugeda, eds., International Centre for Numerical Methods in Engineering (CIMNE), Barcelona, Spain, 1–17.

[8] F. Chalot, G. Chevalier, Q. Dinh, and L. Giraud, (1999), Some investigations of domain decomposition techniques in parallel CFD, In *EUROPAR'99 Parallel Processing*, P. Amestoy, P. Berger, M. Daydé, V. Frayssé, L. Giraud, and D. Ruiz, eds., vol. 1685, Lecture Notes in Computer Science, Springer-Verlag, 595 – 602.

[9] G. Degrez, L. Giraud, M. Loriot, A. Micelotta, B. Nitrosso, and A. Stoessel, (1995), Parallel Industrial CFD calculations with N3S, In *Lecture Notes in Computer Science, High-Performance Computing and Networking*, vol. 919, Springer-Verlag, 820–825.

[10] V. Frayssé, L. Giraud, and H. Kharraz-Aroussi, (1998), On the influence of the orthogonalization scheme on the parallel performance of GMRES, In *EUROPAR'98 Parallel Processing*, D. Pritchard and J. Reeve, eds., vol. 1470, Springer, 751–762.

[11] V. Frayssé, L. Giraud, and V. Toumazou, (1996), Parallel computation of spectral portraits on the Meiko CS2, In *High-Performance Computing and Networking*, H. Liddell, A. Colbrook, B. Hertzberger, and P. Sloot, eds., vol. 1067, Springer-Verlag, 312–318.

[12] L. Giraud and G. M. Manzini, (1994), Parallel Distributed Implementations of 2D Explicit Euler Solver, In *Lecture Notes in Computer Science, High-Performance Computing and Networking*, vol. 796, Springer-Verlag, 151–156.

[13] L. Giraud and P. Spitéri, (1989), Résolution par des algorithmes de relaxation parallèles des équations d'Hamilton-Jacobi-Bellman discrétisées et linéarisées, *Publication Mathématiques de Besançon*, 31–46.

[14] L. Giraud and P. Spitéri, (1992), Implementation of parallel solutions for nonlinear boundary value problems, In *Parallel Computing'91*, G. J. D.J. Evans and H. Liddel, eds., Elsevier Science Publishers, 203–211.

[15] L. Giraud and R. S. Tuminaro, (1993), Domain Decomposition Algorithms for the Drift-Diffusion Equations, In *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing*, R. Sincovec, D. Keyes, M. Leuze, L. Petzold, and D. Reed, eds., SIAM, 719–726.

[16] L. Giraud and R. S. Tuminaro, (1995), Domain Decomposition Algorithms for PDE problems with large scale variations, In *Domain Decomposition Methods in Scientific and Engineering Computing*, D. Keyes and J. Xu, eds., vol. 180, American Mathematical Society - Contemporary Mathematics, 205–210.

[17] L. Giraud, R. Guivarch, and J. Stein, (1999), A parallel distributed fast 3D Poisson solver for MésoNH, In *EUROPAR'99 Parallel Processing*, P. Amestoy, P. Berger, M. Daydé, V. Frayssé, L. Giraud, and D. Ruiz, eds., vol. 1685, Lecture Notes in Computer Science, Springer-Verlag, 1431 – 1434.

[18] L. Giraud, M. Henry, L. Lefebvre, L. Perret, C. Puglisi, and D. Vaucher, (1995), Parallelization of ASTRYD, a software based on a time domain method for solving problems in vibroacoustics, vol. 2, CETIM, 533–538.

[19] L. Giraud, N. Maman, P. Menegazzi, A. Micelotta, and B. Thomas, (1996), Parallel Industrial Incompressible CFD calculations with HPCN3S, In *High-Performance Computing and Networking*, H. Liddell, A. Colbrook, B. Hertzberger, and P. Sloot, eds., vol. 1067, Springer-Verlag, 122–127.

[20] L. Giraud, J. Miellou, and P. Spitéri, (1991), Implementation of domain decomposition methods on shared multiprocessors, In *High Performance Computing II*, M. Durand and F. E. Dabagh, eds., Elsevier Science Publishers, 357–368.

[21] L. Giraud, (1993), Shared and distributed implementations of block preconditioned conjugate gradient methods using domain decomposition on the BBN TC2000, In *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing*, R. Sincovec, D. Keyes, M. Leuze, L. Petzold, and D. Reed, eds., SIAM, 703–710.

[22] P. Jabouille, R. Guivarch, P. Kloos, D. Gazen, N. Gicquel, L. Giraud, N. Asencio, V. Ducrocq, J. Escobar, J.-L. Redelsperger, J. Stein, and J.-P. Pinty, (1999), Parallelization of the French Meteorological Mesoscale Model MésoNH, In *EUROPAR'99 Parallel Processing*, P. Amestoy, P. Berger, M. Daydé, V. Frayssé, L. Giraud, and D. Ruiz, eds., vol. 1685, Lecture Notes in Computer Science, Springer-Verlag, 1417 – 1422.

[23] J. M. Jimenez, L. Matey, J. Garcia, A. Avello, S. Baldini, L. Giraud, and L. Hamel, (1997), On the Use of High-Performance Computing in Multi-Body Analysis for Two-Wheeled Suspension Design, In *Multi-Body Dynamics: Monitoring and Simulation Techniques*, H. Rahnejat and R. Whalley, eds., Mechanical Engineering Publications Limited, 295–304.

[24] J. Miellou, L. Giraud, A. Laouar, and P. Spitéri, (1991), Subdomain decomposition methods with overlapping and asynchronous iterations, In *Progress in partial differential equations : the Metz surveys*, M. Chipot and J. Paulin, eds., vol. 249, Longman Scientific and Technical, Pitman Research Notes in Mathematics, 166–183.

Technical reports

[1] G. Alléon, M. Benzi, and L. Giraud. Sparse approximate inverse preconditioning for dense linear systems arising in computational electromagnetics. Technical Report TR/PA/97/05, CERFACS, Toulouse, France, 1997. Preliminary version of article in Numerical Algorithms, vol. 6, 1997.

[2] S. Baldini, L. Giraud, L. Hamel, J. M. Jimenez, and L. M. Matey. HIPERCOMBATS : a parallel industrial tool for two-wheeler suspensions design. Technical Report TR/PA/97/08, CERFACS, Toulouse, France, 1997. Preliminary version of proceeding in High Performance Computing and Networking, 1997.

[3] S. Baldini, L. Giraud, J. M. Jimenez, L. M. Matey, and J. G. Izaguirre. High performance computing in multi-body system design. Technical Report TR/PA/97/27, CERFACS, Toulouse, France, 1997. Preliminary version of article in Int J. Supercomputer Applications, vol. 2(13), 1999.

[4] Ph. Berger, L. Giraud, and P. Spitéri. Mise en œuvre d'algorithmes parallèles synchrones et asynchrones sur une architecture multi-transputers. Technical Report , IRIT/URA CNRS 1399, Toulouse, France, 1989.

[5] A. Bouras, V. Frayssé, and L. Giraud. A relaxation strategy for inner-outer linear solvers in domain decomposition methods. Technical Report TR/PA/00/17, CERFACS, Toulouse, France, 2000.

[6] B. Carpentieri, I. S. Duff, and L. Giraud. Some sparse pattern selection strategies for robust frobenius norm minimization preconditioners in electromagnetism. Technical Report RAL-TR-2000-009, Rutherford Appleton Laboratory, 2000. Preliminary version of the paper to appear in Numerical Linear Algebra with Applications.

[7] B. Carpentieri, I.S. Duff, and L.Giraud. Experiments with sparse preconditioning of dense problems from electromagnetic applications. Technical Report TR/PA/00/04, CERFACS, Toulouse, France, 2000.

[8] L.M. Carvalho, I. S. Duff, and L. Giraud. Linear algebra kernels for parallel domain decomposition methods. Technical Report TR/PA/95/26, CERFACS, Toulouse, France, 1995. Preliminary version of proceeding in Advanced Computational Methods in Structural Mechanics, 1996.

[9] L.M. Carvalho and L. Giraud. Additive Schwarz for the Schur complement method. Technical Report TR/PA/96/51, CERFACS, Toulouse, France, September 1996. Preliminary version of proceeding in Domain Decomposition Methods in Scientific Computing, 1998.

[10] L.M. Carvalho and L. Giraud. Block diagonal preconditioners for the Schur complement method. Tech. Rep. TR/PA/97/46, CERFACS, France, 1997. Preliminary version of the Saxe-coburg publication, 1999.

[11] L.M. Carvalho and L. Giraud. Parallel subdomain-based preconditioner for the Schur complement. Technical Report TR/PA/99/04, CERFACS, Toulouse, France, 1999. Preliminary version of the proceeding EUROPAR'99 Parallel Processing, 1999.

[12] L.M. Carvalho, L. Giraud, and G. Meurant. Local preconditioners for two-level non-overlapping domain decomposition methods. Tech. Rep. TR/PA/99/38, CERFACS, France, 1999. Preliminary version of the paper to appear in Numerical Linear Algebra with Applications.

[13] L.M. Carvalho, L. Giraud, and P. Le Tallec. Algebraic two-level preconditioners for the Schur complement method. Tech. Rep. TR/PA/98/18, CERFACS, France, 1998. Preliminary version of the paper to appear in SIAM SISC.

[14] V. Frayssé and L. Giraud. Comparative study of QMR versus block QMR for J-symmetric matrices in electromagnetism applications. Tech. Rep. TR/PA/98/11, CERFACS, Toulouse, France, 1998.

[15] V. Frayssé and L. Giraud. A set of conjugate gradient routines for real and complex arithmetics. Technical Report TR/PA/00/47, CERFACS, Toulouse, France, 2000.

[16] V. Frayssé, L. Giraud, and S. Gratton. A set of GMRES routines for real and complex arithmetics. Tech. Rep. TR/PA/97/49, CERFACS, 1997.

[17] V. Frayssé, L. Giraud, and S. Gratton. A set of flexible-GMRES routines for real and complex arithmetics. Tech. Rep. TR/PA/98/20, CERFACS, France, 1998.

[18] V. Frayssé, L. Giraud, and H. Kharraz-Aroussi. On the influence of the orthogonalization scheme on the parallel performance of GMRES. Technical Report TR/PA/98/07, CERFACS, 1998.

[19] V. Frayssé, L. Giraud, and V. Toumazou. Parallel computation of spectral portraits on the Meiko CS2. Technical Report TR/PA/96/02, CERFACS, Toulouse, France, 1996. Preliminary version of proceeding in High-Performance Computing and Networking, 1996.

[20] L. Giraud. *Implantations parallèles de méthodes de sous-domaines synchrones et asynchrones pour la résolution de problèmes aux limites.* PhD thesis, Institut National Polytechnique de Toulouse, 1991.

[21] L. Giraud. Shared and distributed implementations of block preconditioned conjugate gradient using domain decomposition on a distributed virtual shared memory computer. Tech. Rep. TR/PA/92/91, CERFACS, Toulouse, France, 1992.

[22] L. Giraud, D. Lugato, and F. Saab. Parallel distributed fast 3D Poisson solver. Tech. Rep. TR/PA/97/58, CERFACS, Toulouse, France, 1997.

[23] L. Giraud and G. M. Manzini. Parallel implementations of a multidomain explicit high-order accurate Euler solver. Tech. Rep. TR/CFD-PA/93/49, CERFACS, Toulouse, France, 1993.

[24] L. Giraud, P. Noyret, E. Sevault, and V. Van Kemenade. IPM - user's guide and reference manual. Tech. Rep. TR/PA/95/01, CERFACS, Toulouse, France, 1995.

[25] L. Giraud and P. Spitéri. Résolution parallèle des équations d'Hamilton-Jacobi-Bellman sur un calculateur distribué. Technical Report , IRIT/URA CNRS 1399, Toulouse, France, 1989.

[26] L. Giraud and P. Spitéri. Implantation d'algorithmes de relaxation parallèles synchrones et asynchrones sur l'architecture CAPITAN-MATRA. Technical Report , IRIT/URA CNRS 1399, Toulouse, France, 1990.

[27] L. Giraud and P. Spitéri. Résolution parallèle de problèmes d'équations aux déquations aux dérivées partielles non-linéaires sur une architecture à mémoire distribuée. Technical Report IRIT/91-1-R, IRIT/URA CNRS 1399, Toulouse, France, 1991.

[28] L. Giraud, P. Spitéri, and J.C. Miellou. Méthodes de gradient conjugué 3D préconditionnées par un solveur 2D de type méthode alternée de Schwarz. Technical Report IRIT/91-6-R, IRIT/URA CNRS 1399, Toulouse, France, 1991.

[29] L. Giraud and R. S. Tuminaro. A domain decomposition probing variant suitable for anisotropic problems. Tech. Rep. TR/PA/93/36, CERFACS, Toulouse, France, 1993.

[30] L. Giraud and R. S. Tuminaro. Grid transfer operators for highly variable coefficient problems. Tech. Rep. TR/PA/93/37, CERFACS, Toulouse, France, 1993.

[31] L. Giraud and R. S. Tuminaro. Time dependent solvers on distributed memory computers. Tech. Rep. TR/PA/95/01, CERFACS, Toulouse, France, 1995. Preliminary version of article in Calculateurs parallèles, vol. 7(3), 1995.

[32] L. Giraud and R. S. Tuminaro. Schur complement preconditioners for anisotropic problems. Tech. Rep. 98-8488J, Sandia National Laboratories, 1998. Preliminary version of article in IMA J. Numerical Analysis, vol. 19 (1), 1999.

Contract reports

The list below does not include any report deliverables from the European projects I have been involved in.

[1] C. Amestoy, M. J. Daydé, and L. Giraud, (1994a), Parallélisation conjointe CERFACS-CNES d'un code pilote du CNES, Final Contract Report FR/PA/94/13, CERFACS, Toulouse, France.

[2] C. Amestoy, M. J. Daydé, and L. Giraud, (1994b), Parallélisation conjointe CERFACS-CNES d'un code pilote du CNES: Bilan de la première phase, Intermediate Report IR/PA/94/06, CERFACS, Toulouse, France.

[3] M. Benzi and L. Giraud, (1997), Acquisition d'une méthode pour la résolution des systèmes linéaires issus des problèmes d'électromagnétisme, Final Contract Report FR/PA/97/04, CERFACS, Toulouse, France.

[4] F. Chalot, G. Chevalier, Q. V. Dinh, and L. Giraud, (1999), Some investigations of domain decomposition techniques in parallel CFD, Tech. Rep. TR/CFD/99/06, CERFACS, Toulouse, France. Preliminary version of the proceeding EUROPAR'99 Parallel Processing, 1999.

[5] F. Collino, S. Ghanemi, L. Giraud, S. Gratton, M. Invernizzi, P. Joly, and A. Piacentini, (1995), Rapport de fin de contrat Thomson: Résolution des équations de Maxwell tridimensionnelles dans le domaine fréquentiel sur réseaux hétérogènes de calculateurs, Final Contract Report FR/EL-PA/95/25, CERFACS, Toulouse, France.

[6] M. J. Daydé, I. S. Duff, J. Y. L'Excellent, and L. Giraud, (1992), Evaluation d'ordinateurs vectoriels et parallèles sur un jeu de programmes représentatifs des calculs intensifs à la division avions de l'Aérospatiale : bilan de l'étape de portage, Preliminary Report PR/PA/92/10, CERFACS, Toulouse, France.

[7] M. J. Daydé, I. S. Duff, J. Y. L'Excellent, and L. Giraud, (1993), Evaluation d'ordinateurs vectoriels et parallèles sur un jeu de programmes représentatifs des calculs intensifs à la division Avions de l'Aerospatiale, Final Contract Report PR/PA/93/19, CERFACS, Toulouse, France.

[8] V. Frayssé and L. Giraud, (1997), An implementation of block QMR for J-symmetric matrices, Final Contract Report FR/PA/97/57, CERFACS, Toulouse, France.

[9] V. Frayssé, L. Giraud, and S. Gratton, (1996), Solveurs linéaires itératifs pour la résolution de systèmes complexes non hermitiens creux de grande taille, Final Contract Report FR/PA/96/34, CERFACS, Toulouse, France.

[10] L. Giraud and S. Gratton, (1995), Solveurs linéaires performants pour la résolution de systèmes complexes non hermitiens creux de grande taille, Final Contract Report TR/PA/95/24, CERFACS, Toulouse, France.

Appendix: resume in French

Luc Giraud

Adresse professionnelle
CERFACS
42 av. Gaspard Coriolis
31057 Toulouse cedex 01
Tél: 05 61 19 30 25
Fax: 05 61 19 30 00
Email: `giraud@cerfacs.fr`
Internet: `http://www.cerfacs.fr/~giraud`

Adresse personnelle
6 rue des Sabots
31400 Toulouse
Tél: 05 61 25 89 52

Né le 16 Juillet 1965

FORMATION

1992 **Qualification** aux fonctions de maître de conférence en $26^{\text{ème}}$ et $27^{\text{ème}}$ section.

1991 **Doctorat** en Informatique et Mathématiques Appliquées de l'Institut National Polytechnique de Toulouse.

1988 **Diplôme d'Ingénieur** en Informatique et Mathématiques Appliquées de l'**ENSEEIHT**.

1988 **D**iplôme d'**E**tudes **A**pprofondies en Informatique, ENSEEIHT, mention Bien.

1983-1985 Classes préparatoires, lycée Dumont-Durville, Toulon.

1983 Baccalauréat Série E, mention Bien.

EXPERIENCE PROFESSIONNELLE

Depuis Octobre 1993, **Chercheur Senior** dans l'équipe Algorithmes Parallèles du CERFACS (Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique), Toulouse.

D'Octobre 1991 à Octobre 1993, **Chercheur post-doctoral** au CERFACS.

De Septembre 1988 à Mai 1991, **Thèse de Doctorat INPT en informatique** sur l'étude et l'implantation de méthodes de sous-domaines synchrones et asynchrones pour la résolution de problèmes aux limites préparée à l'ENSEEIHT (École Nationale Supérieure d'Electronique, d'Electrotechnique, d'Informatique et d'Hydraulique de Toulouse).
Moniteur en informatique à l'ENSEEIHT durant l'année universitaire 1989-1990.

LANGUES

Anglais lu, écrit et parlé couramment. Langue de travail.

ACTIVITÉS DE RECHERCHE

PROBLÉMATIQUE

L'évolution permanente de l'architecture des calculateurs scientifiques et des paradigmes de programmation associés nécessitent la remise en cause à la fois des algorithmes utilisés en simulation numérique et de leur implantation. C'est dans ce contexte que s'est développée ma recherche, plus spécifiquement focalisée sur des techniques de résolution itérative de systèmes linéaires denses et creux de grande taille pour des architectures de calculateurs multiprocesseur à mémoire partagée et distribuée. Je me suis notamment intéressé à la résolution de systèmes linéaires issus de la discrétisation de problèmes aux dérivées partielles et notamment aux méthodes de décomposition de domaines particulièrement adaptées à l'architecture des calculateurs à mémoire distribuée. Mon travail dans ce contexte consiste en l'étude du comportement numérique de nouveaux préconditionneurs et de l'implantation de ces techniques sur des architectures parallèles pour en vérifier leur pertinence d'un point de vue performance informatique. Comme exemples d'applications dans lesquelles j'ai pu mettre en œuvre ce type de méthodes, je citerai l'aérodymanique, la météorologie ou la simulation de composants semi-conducteurs.

COLLABORATIONS

- **Internationales**

  Depuis 1994 Collaboration avec R. Tuminaro, chercheur à Sandia National Laboratories, Livermore (U.S.A), sur des techniques de décomposition de domaines et de préconditionnement.

  1996-1997 Responsable pour le CERFACS de la collaboration CERFACS-CRS4 dans le cadre des projets bilatéraux franco-italien Gallilée. Le thème de cette collaboration était l'utilisation de techniques d'algèbre linéaire parallèles en simulation numérique de pollution des sols par des contaminants chimiques.

  1994-1996 Participant au projet Esprit HCM intitulé **Iterative Methods**. Ce réseau de recherche coordonné par le CERFACS avait pour objectif l'étude de nouvelles techniques pour la résolution itérative de systèmes linéaires creux de grande taille avec une attention particulière portée au cas des systèmes linéaires non-symétriques. Les autres partenaires de ce réseau de recherche étaient le CEA (France), le CRS4 (Italie), Dassault Aviation (France), IAN-CNR (Italie) et Utrecht University (Pays Bas).

  1994-1997 Participant au projet Esprit HCM intitulé **Advanced finite element techniques** IAN-CNR (Italie), National Technical University of Athens (Grèce), Instito Superior Tecnico (Portugal), Universitat Politecnica de Catalunya (Espagne), Heriot-Wat University (Angleterre), University of Wales (Angleterre), University of Essen (Allemagne).

- **Nationales**

  Depuis 1996 Responsable d'une collaboration avec l'INRIA-Rocquencourt sur le thème de la décomposition de domaine avec application à la simulation numérique de semiconducteurs. Les correspondants INRIA sur cette collaboration sont P. Le Tallec et A. Marrocco.

  Depuis 1991 Collaboration avec les chercheurs du Projet Algorithmes Parallèles et ceux des Projets applicatifs du CERFACS.

RÉALISATIONS LOGICIELLES

Mon travail est indissociable d'une pratique logicielle qui conduit à des développements d'outils pour la simulation numérique. Certaines des implantations logicielles sur lesquelles j'ai travaillé ont pu être placées dans le domaine public à la disposition de la communauté scientifique à des fins non-commerciales. Ces outils logiciels ont été conçus pour être exploités indifféremment sur des calculateurs scalaires ou vectoriels ou sur des multiprocesseurs à mémoire partagée ou distribuée. Les codes source ainsi que les manuels d'utilisation sont référencés et accessibles sur le Web:

`http://www.cerfacs.fr/algor/Softs/`

2000 "A Set of Conjugate Gradient Routines for Real and Complex Arithmetics" réalisé en collaboration avec V. Frayssé.

1998 "A Set of Flexible-GMRES Routines for Real and Complex Arithmetics" réalisé en collaboration avec V. Frayssé et S. Gratton.

1997 "A Set of GMRES Routines for Real and Complex Arithmetics" réalisé en collaboration avec V. Frayssé et S. Gratton. Cet ensemble logiciel est en cours d'acquisistion par une société de services allemande pour son intégration dans un logiciel commercialisé de calcul de structures.

ENCADREMENT SCIENTIFIQUE

En tant que chercheur senior dans le Projet Algorithmes Parallèles du CERFACS depuis 1993, je participe à la coordination scientifique quotidienne d'une équipe d'une quinzaine de chercheurs (doctorants et post-doctoraux), sous la responsabilité de Iain S. Duff. Par ailleurs, le Projet Algorithmes Parallèles accueille régulièrement des étudiants ENSEEIHT de la filière informatique et mathématiques appliquèes dont le stage se déroule sur toute l'année universitaire à raison de trois jours par semaine au CERFACS.

De façon plus spécifique, j'ai eu à encadrer directement les personnes suivantes:

Post-docs et ingénieurs d'études

1997-1999 Ph. Kloos, ingénieur d'étude CERFACS, qui a travaillé durant deux années dans le groupe Méso-NH du CERFACS.

1998 R. Guivarch, qui, en tant que post-doctorant au sein du groupe de travail Méso-NH, a contribué au développement de la bibliothèque de gestion transparente du parallélisme.

1996-1998 B. Hamma, qui a travaillé pendant deux ans sur les sujets liés au projet ODESIM (parallélisme et optimisation).

1995 M.M. Magolu, qui, en tant que post-doctorant, a travaillé sur la parallélisation du solveur linéaire du code N3S dans le cadre du projet Esprit HPCN3S.

Doctorants

Depuis Octobre 1999 grâce à une dérogation de l'école doctorale de Mathématiques appliquées de Toulouse j'assure la direction scientifique de la thèse de J. Langou. Le sujet de cette thèse financée par Aerospatiale est le développement de solveur itèratifs multi-second membres pour la résolution de systèmes linéaires en acoustique et électromagnétisme issus d'une formulation par équation intégrale.

Depuis Février 1999 j'assure la direction scientifique de la thèse de J.-C. Rioual en collaboration avec P. Amestoy (ENSEEIHT). Le sujet de cette thèse est le développement de préconditionneurs parallèles et robustes pour les systèmes linéaires en simulation de semi-conducteurs. Ce travail de thèse s'inscrit dans le cadre de la collaboration avec l'INRIA.

Depuis 1998 conjointement avec I. S. Duff, j'assure l'encadrement scientifique des travaux de B. Carpentieri qui travaille sur des techniques de préconditionnement pour la résolution itérative de systèmes linéaires complexes denses de grande taille.

1994-1997 j'ai assuré la direction scientifique de la thèse de L. Carvalho qui était consacrée à l'étude et l'implantation parallèle de préconditionneurs à deux niveaux pour des méthodes de décomposition de domaines sans recouvrement. L. Carvalho a soutenu sa thèse en Octobre 1997 devant le jury composé de P. Le Tallec (Président et rapporteur), G. Meurant (Rapporteur), I.S. Duff, N. Maculan, J. Noailles, et moi-même.

Stagiaires 3$^{\text{ème}}$ cycle

1999-2000 F. Guevara, élève 3$^{\text{ème}}$ année ENSEEIHT + DEA, travaille sur des techniques de préconditionnement multi-niveau pour les techniques de décomposition de domaines sans-recouvrement.

1997-1998 D. Lugato, élève 3$^{\text{ème}}$ année ENSEEIHT, qui a été intégré dans l'équipe travaillant sur le développement de la bibliothèque de gestion du parallélisme pour le projet Méso-NH.

1997-1998 G. Torres, élève 3$^{\text{ème}}$ année ENSEEIHT + DEA, qui a parallélisé, via des techniques de sous-domaines sans recouvrement, un code INRIA de simulation de semiconducteurs (éléments finis mixtes non-structurés). Il a en particulier étudié le comportement numérique de différents préconditionneurs locaux pour le complément de Schur. Ses réalisations ont servi de point de départ pour le travail de thèse de J.-C. Rioual.

1997-1998 B. Thomas, élève 3$^{\text{ème}}$ année ENSEEIHT, a travaillé sur le portage sur PC sous Windows-NT et sur l'amélioration du gestionnaire de tâches parallèles écrit par L. Hamel. Ces travaux ont été en partie présentés au cours d'une communication à EuroPar'99.

1997-1998 Y. Thiaudière, élève 3$^{\text{ème}}$ année ENSEEIHT, a enrichi le gestionnaire de tâches parallèles sur réseau de stations Unix en intégrant la possibilité de gérer deux niveaux de parallélisme (tâches principales générant des sous-tâches). Ces travaux ont été en partie présentés au cours d'une communication à EuroPar'99.

1996-1997 F. Saab, élève 3$^{\text{ème}}$ année ENSEEIHT, qui a travaillé sur différentes stratégie de parallélisation d'un solveur de Poisson rapide sur machines à mémoire distribuée.

1995-1996 L. Hamel, élève 3$^{\text{ème}}$ année ENSEEIHT + DEA qui a travaillé sur un sujet traitant de calcul hétérogène sur réseau de stations Unix et a réalisé un gestionnaire de tâches parallèles. Ces travaux ont été en partie présentés au cours d'une communication à HPCN'96.

Stagiaires 2$^{\text{nd}}$ cycle

Juillet - Septembre 1999 J.M. Donaville a effectué son stage de 2$^{\text{ème}}$ année ENSIMAG sur l'étude d'une variante de la méthode alternée de Schwarz utilée comme préconditionneur.

Juillet - Septembre 1999 G. Lartigue a effectué son stage de 2$^{\text{ème}}$ année ENSEEIHT, filière hydraulique sur un sujet que j'ai proposé et dont l'encadrement était assuré en collaboration avec un post-doctorant de l'équipe CFD du CERFACS. Ce sujet portait sur un maquettage d'une variante de la méthode de Schur avec solveurs locaux inexacts en vue de son implantation dans un code industriel Navier-Stokes 3D non-structuré.

ORGANISATION DE CONFÉRENCES

- Membre du comité scientifique des conférences VecPar'96, VecPar'98 et VecPar'2000.

- **2000** Membre du comité scientifique de la conférence HPCN'2000.

- **1999** Membre du comité d'organisation d'EuroPar'99 conjointement organisé par le CERFACS et l'ENSEEIHT. Local-chair d'un topic dédié aux projets européens et co-éditeur des proceedings de la conférence publié par Springer Verlag dans la série "Lecture notes in computer science".

- **1997** Membre du comité scientifique de la conférence PVM-MPI Europe'97.

- **Sept. 1995 - Sept. 1996** Membre du comité d'organisation de l'"International Linear Algebra Year" (série de 4 workshops) organisée par le CERFACS.

- **Juin 96** Co-organisation avec C. Douglas (IBM Yorktown et Yale University) du workshop "Iterative Methods" dans le cadre de l'"International Linear Algebra Year" et "guest editor" de la sélection des papiers de ce workshop publiés dans BIT.

- **Mai 1996** Organisation du workshop ODESIM auquel étaient invités les industriels potentiellement intéressés par l'outil logiciel résultant du projet.

- **Sept. 1996** Co-organisation avec I.S. Duff d'un mini-symposium à ECCOMAS'96 intitulé "Scientific computing at CERFACS".

REFEREE

Pour des revues scientifiques internationales : Computer Physics Communications, Journal of Computational Physics, Int. J. of Supercomputer Applic. and High Perf. Comp., SIAM J. Sci. Comp., BIT.

Pour des conférences internationales : VecPar, PVM-MPI, Copper Mountain, HPCN, ConPar V, Civil-Comp, Leslie Fox Price.

Expert auprès de la Commission Européenne pour les projets "Long Term Research" du $4^{\text{ème}}$ PCRD et les projets "Technologie de l'Information" du $5^{\text{ème}}$ PCRD.

ACTIVITÉS CONTRACTUELLES

La recherche de financements, et le suivi technique, administratif et budgétaire de collaborations contractuelles avec l'industrie sont des activités fortement encouragées au CERFACS, dans le but d'obtenir des ressources extérieures permettant d'accroître le potentiel humain et matériel consacré aux recherches plus amont. Dans la mesure du possible, nous nous efforçons de concevoir des collaborations industrielles dont le contenu scientifique exploite et voire fait progresser l'état de nos recherches.

1999-2002 Responsable scientifique de la collaboration Aerospatiale CCR (Centre Commun de Recherches) sur "l'étude de solveurs parallèles robustes et efficaces pour la résolution de systèmes linéaires avec seconds membres multiples en électromagnétisme et acoustique exploitant la méthode multipôle". Cette collaboration est mise en place au travers d'une thèse financée par Aerospatiale et encadrée conjointement par le CERFACS et Aerospatiale.

1996-2000 Responsable scientifique et animateur du groupe de travail CERFACS dans le cadre de la collaboration avec Météo-France pour la "Parallélisation du code Méso-NH sur machines à mémoire distribuée". Ce code de recherche en météorologie méso-échelle est développé conjointement par le CNRM (Centre National de Recherches de Météo-France) et le Laboratoire d'Aérologie. Dans ce contexte une bibliothèque encapsulant les échanges de messages et masquant l'utilisation du parallélisme à été spécifée puis implantée. Par ailleurs, diverses stratégies de parallélisation ont été étudiées afin de porter le solveur de Poisson rapide implanté comme préconditionneur dans le solveur de Pression.
Les résultats de ces travaux ont fait l'objet de deux communications à Euro-Par'99 et d'une communication au colloque d'analysse numérque CANUM'99.

1998-1999 Co-ordinateur du projet Esprit PST intitulé **MYSHANET** pour "Parallel Multibody simulation for shock absorber design on PC network". Co-ordonné par le CERFACS, ce projet s'inscrivait dans le cadre des projets européens de transfert de technologie vers les PME/PMI. L'objectif de ce projet était de porter sur réseau de PC sous windows le code HIPERCOMBATS en vue de son utilisation pour le conception d'amortisseurs. Les autres partenaires de ce consortium étaient : le CEIT (centre de recherches, Espagne), Donerre Amortisseur (industriel, France) et Marzocchi (industriel, Italie).
Les résultats de ces travaux ont fait l'objet d'une communication à Euro-Par'99.

1998-1999 Participant à une collaboration avec Dassault Aviation dont le but était d'étudier des préconditioneurs parallèles pour accélérer la convergence des systèmes linéaires intervenant dans le schéma implicite implanté dans un code Navier-Stokes tridimensionnel éléments finis non-structurés. Cette étude a été réalisée en collaboration avec l'équipe CFD (Computational Fluid Dynamic) du CERFACS qui en avait la responsabilité.
Les résultats de ces travaux ont fait l'objet de deux communications, une à Euro-Par'99 et une à Parallel CFD'99.

1996-1998 Responsable CERFACS du projet Esprit HPCN intitulé **ODESIM** signifiant "Optimum DESIgn of Multi-body sytems". Ce projet était coordonné par le CEIT avec la participation de CASA (industriel, Espagne), CR Fiat (industriel, Italie), CERFACS, Matra-Datavision (industriel, France) et Siemens (industriel, Allemagne). Le but de ce projet était de montrer que des outils de simulation multi-corps et de CAO pouvaient être intégrés afin de permettre de l'optimisation de mécanismes multicorps en utilisant les ressources de calcul de réseaux hétérogènes de stations de travail. La contribution du CERFACS concernait à la fois dans le choix des techniques numériques en optimisation ainsi que la gestion de tâches parallèles sur réseau hétérogène de stations de travail en mode non-dédié.

1997 Responsable CERFACS du contrat Aerospatiale CCR portant sur l'"étude et l'implantation d'une variante de la méthode Block-QMR pour matrices J-symétriques".

1996 Responsable CERFACS du contrat Aerospatiale CCR: "Acquisition d'une méthode pour la résolution des systèmes linéaires issus des problèmes d'électromagnétisme". L'objectif de cet contrat était d'étudier l'utilisation de technique de préconditionnement par inverse approchée appliquées à des systèmes linéaires denses complexe symétriques non-hermitiens pour la résolution de problèmes d'électromagnétisme en formulation intégrale.
Les résultats de ce travail a fait l'objet d'un article publié dans Numerical Algorithms.

1996 Participant au contrat CNES: "Solveurs linéaires itératifs pour la résolution de systèmes complexes non hermitiens creux de grande taille". Ce projet visait à développer un solveur de type GMRES en arithmétique complexe.

1995-1996 Responsable CERFACS du projet Esprit CAPRI intitulé **HIPERCOMBATS** signifiant "HIgh PERformance COmputing in Multi-Body Analysis for Two-wheeler Suspension design". Ce projet était coordonné par Piaggio (industriel, Italie) avec la participation du CERFACS et du CEIT. L'objectif de ce projet était de développer un outil parallèle sur réseau de stations Unix pour la simulation paramétrique de multi-corps articulés. Le rôle du CERFACS était de développer un module de gestion de tâches parallèles sur réseau hétérogène non-dédié ayant des fonctionalités de gestion dynamique de la charge et de tolérance aux pannes.
Les résultats de ces travaux ont fait l'objet de deux communications, la première à HPCN'96, la seconde une à Multi-Body Dynamics: Monitoring and Simulation Techniques'97.

1995 Responsable CERFACS du contrat avec Thomson LCR : "Résolution des équations de Maxwell tridimensionnelles dans le domaine fréquentiel sur réseaux hétérogènes de calculateurs". Cette étude portait sur la résolution itérative de systèmes linéaires issus de discrétisation par éléments finis pour la résolution parallèles des equations de Maxwell.

1994-1996 Responsable CERFACS du projet Esprit HPCN intitulé **HPCN3S** signifiant "High Perfomance Computing and Networking with N3S". Ce projet était coordonné par Simulog (industriel, France) avec la participation du CERFACS, CISE (industriel, Italie), EDF (centre de recherche, France), IFP (industriel, France), VKI (centre de recherche, Belgique) et visait à porter, sur machines à mémoire distribuée, le code de mécanique des fluides compressible et incompressible N3S développé par EDF et l'INRIA. La contribution CERFACS était principalement la parallélisation du solveur linéaire du code incompressible.
Les résultats de ces travaux ont fait l'objet de deux communications l'une à HPCN'95 et l'autre à HPCN'96.

1993-1994 Responsable CERFACS du contrat CNES intitulé "Parallèlisation d'un programme CNES dans le cadre du développement de l'activité calcul parallèle au CNES". Cette étude était composée d'une partie expertise d'un code d'acoustique en vue de sa parallèlisation et d'une composante formation des ingénieurs du CNES appelés à developper l'activité calcul parallèle au sein du CNES.
Ces travaux ont été présentés lors d'une communication à CETIM'95.

1991-1992 Participant à la réalisation du contrat avec Aerospatiale Division Avions: "Evaluation d'ordinateurs vectoriels et parallèles sur un jeu de programmes représentatifs des calculs intensifs à la division avions d'Aerospatiale". Dans le cadre de ce travail, nous avons été amené à adapter puis évaluer les performances de certains codes représentatifs de l'activité calcul scientifique Aerospatiale sur une large gamme de calculateurs hautes performances incluant des machines parallèles vectorielles à mémoire partagée, les premières machines à mémoire distribuée virtuellement partagée ainsi que des réseaux de stations de travail.

ACTIVITÉS D'ENSEIGNEMENT ET DE FORMATION

L'ensemble des formations et enseignements que j'ai pu dispenser l'ont été essentiellement sur le calcul scientifique parallèle autour de deux grands thèmes qui sont le calcul hautes performances et l'algèbre linéaire sur calculateurs parallèles. Les formations/cours sur le calcul hautes performances couvrent des aspects principalement informatiques incluant la présentation des architectures des

calculateurs scientifiques et les mécanismes de base mis en œuvre au niveau matériel, ainsi que la présentation des outils et environnements de programmations disponibles sur ces plateformes. Ces notions sont également présentes dans les cours relatifs à l'algèbre linéaire, qui sont à dominantes numériques et visent à illustrer comment le choix d'une méthode numérique et son comportement sont intimement liés aux spécificités du calculateur cible sur lequel elle sera exploitée. Ci-dessous est listé l'ensemble des cours/formations auxquelles j'ai participé.

Mars 2000 Co-organisation avec V. Frayssé d'une formation de trois jours intitulée "Outils de programmation efficace et robuste pour le logiciel scientifique" dispensée à un groupe d'ingénieurs et chercheurs du CNES.

Depuis 1994 Cours (12 h) dans le module Calcul Parallèle de l'option de troisième année ENSICA (Ecole Nationale Supérieure d'Ingénieurs en Construction Aéronautique - Toulouse). Dans ce cours je traite essentiellement des méthodes itératives pour la résolution de problèmes d'EDP ainsi que de leur mise en œuvre sur des multiprocesseurs à mémoire partagée et distribuée.

1996 Cours (3 h) d'introduction au calcul hautes performances dans le cadre du Mastère de Météorologie de Météo-France.

Depuis 1997, j'interviens chaque année dans le Mastère de Météorologie de Météo-France. Avec V. Frayssé et B. Cuenot, nous avons conçu un enseignement de calcul scientifique (cours et travaux pratiques) qui part de l'équation différentielle discrétisée par éléments finis pour arriver à sa résolution parallèle. J'interviens dans cette formation pour 5 heures de cours et 3 heures de travaux pratiques.

Depuis 1996 Co-organisation d'une formation interne au CERFACS à l'intention des nouveaux thésitifs et post-doctorants. Le but de cette formation est de leur présenter les architectures des calculateurs hautes performances ainsi que les outils logiciels (bibiothèques d'échange de messages et OpenMP) permettant de les programmer efficacement.

Sept. 1999 Conférencier invité dans le cadre de la "Première école d'été en calcul numérique et symbolique de Rabat". Le thème de l'exposé (2 h) était la résolution parallèle de systèmes linéaires via des méthodes itératives.

Sept. 1999 Intervenant et organisateur avec F. Desprez (ENSL-INRIA) dans le tutorial MPI-OpenMP organisé dans le cadre d'EuroPar'99.

1995-1999 Cours (16 h) de tronc commun dans le cadre du DEA de mathématiques appliquées de l'ENSAE/INSA/UPS. Dans ce cours je traitais essentiellement des méthodes itératives pour la résolution de problèmes d'EDP ainsi que de leur mise en œuvre sur des multiprocesseurs à mémoire partagée et distribuée.
Le mode de fonctionnement de ce DEA est tel que les thèmes des cours sont renouvelés tous les 5 ans. A l'issu de cette période et à compter de la rentrée 1999-2000, ce cours a été intégré dans le cursus de la dernière année INSA spécialité "Génie Mathématique et Modélisation".

Juillet 1998 Assistant à l'école d'été CEA/EDF/INRIA consacré au calcul parallèle.

Juin 1994 Conférencier invité dans le cadre du cours intitulé "Parallélisation de grands codes, applications industrielles et à la recherche" organisé conjointement par la SMAI et le CNRS à l'IDRIS.

Avril 1994 Co-organisateur de deux journées de formation au CERFACS intitulées "Calcul Distribué sur Réseaux de Station de Travail" avec le support d'Aerospatiale Division Avions.

Les cours dispensés durant de ces deux journées étaient destinés aux ingénieurs des organismes partenaires du CERFACS ou ayant des collaborations avec le CERFACS. En plus de la définition et de l'encadrement de la journée consacrée aux travaux pratiques j'ai fait deux présentations l'une sur les outils logiciels disponible (PVM, P4, ...), l'autre sur des exemples de mise en œuvre dans des codes industriels.

1992-1993 Cours COMETT CERFACS/EPFL/INPT intitulé "Computation in Sciences, Methods and Algorithms on Supercomputing for Engineering (COSMASE)". Les diffférents cours portaient sur les concepts de base pour les machines hautes performances, les architectures des machines à mémoire distribuée, les bibliothèques d'échanges de messages, des expériences de parallélisation de codes industriels.

1992-1993 Formation (introduction aux architectures distribuées et bibliothèques d'échanges de messages) dans le cadre du projet européen RECITE du programme FEDER (Toulouse, Bilbao, 1992 - Valence, 1993)

1991 Cours (12 h) à l'ESSI (Sophia-Antipolis) sur des algorithmes parallèles en algèbre linéaire.

1991 TD et TP en 3$^{\text{ème}}$ Année Informatique, Section Parallélisme, à l'ENSEEIHT.

1989-1990 En tant que moniteur de l'enseignement supérieur à l'Université Paul Sabatier de Toulouse, j'ai effectué mon service d'enseignement à l'ENSEEIHT. En année Spéciale Informatique, j'ai assuré des travaux dirigés de compilation, théorie des langages et de systèmes opératoires, ainsi que les travaux pratiques associés.

1989-1990 TP d'informatique en classes préparatoires, Mathématiques supérieures, au lycée Pierre de Fermat à Toulouse.

RESPONSABILITÉS ADMINISTRATIVES

Une des caractéristiques originales du CERFACS est d'avoir choisi des Chefs de Projet dont l'activité principale n'est pas au CERFACS, et qui ne sont donc présents qu'à temps partiel. Les chercheurs seniors ont donc un rôle très important à jouer dans l'organisation et la gestion des équipes de recherche.
Depuis Octobre 1993, je suis l'un des deux chercheurs seniors (permanents) dans le Projet Algorithmiques Parallèles dirigée par I. S. Duff et qui comprend une quinzaine de chercheurs non-permanents (thésitifs et post-doctorants). Mes responsabilités au sein de cette équipe comprennent la gestion administrative et budgétaire au quotidien incluant la gestion de contrats industriels, achat/renouvellement de l'équipement informatique, la rédaction de demandes d'heures de calcul dans différents centres français ou européens, le recrutement, la représentation de l'équipe, le suivi des collaborations avec les autres projets de recherche du CERFACS, la participation à la rédaction de propositions de réponse à différents appels d'offres européens, ...