N^o D'ORDRE: 2244 CERFACS Report: TH/PA/05/57

THÈSE

présentée en vue de l'obtention du titre de

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE TOULOUSE

Spécialité: Informatique

par

Emeric Martin

CERFACS

Préconditionneurs spectraux deux niveaux pour des systèmes linéaires donnés en séquence

Spectral two-level preconditioners for sequences of linear systems

---

Thèse présentée le 26 juillet 2005 à Toulouse devant le jury composé de:

Guillaume Alléon	Directeur de groupe, EADS–CCR	Invité
Abderrahmane Bendali	Professeur, INSA-Toulouse	Président du jury
Angelika Bunse-Gerstner	Professeur, Université de Brême	Rapporteur
Iain S. Duff	Professeur, CERFACS et RAL	Membre du jury
Luc Giraud	Chercheur sénior, CERFACS	Directeur de thèse
Stéphane Lanteri	Directeur de recherche, INRIA	Rapporteur
Gérard Meurant	Directeur de recherche, CEA	Membre du jury

Résumé

De nombreuses simulations numériques nécessitent la résolution d'une série de systèmes linéaires impliquant une même matrice mais des second-membres différents. Des méthodes efficaces pour ce type de problèmes cherchent à tirer bénéfice des résolutions précédentes pour accélérer les résolutions restantes. Deux grandes classes se distinguent dans la façon de procéder: la première vise à réutiliser une partie du sous-espace de Krylov, la deuxième à construire une mise à jour du préconditionneur à partir de vecteurs approximant un espace invariant. Dans cette thèse, nous nous sommes intéressés à cette dernière approche en cherchant à améliorer le préconditionneur d'origine.

Dans une première partie, une seule mise à jour du préconditionneur est considérée pour tous les systèmes. Cette mise à jour consiste en une correction spectrale de rang faible qui permet de translater de un la position des plus petites valeurs propres en module de la matrice du système préconditionné de départ. Des expérimentations numériques sont réalisées en utilisant la méthode GMRES couplée à un préconditionneur de type inverse approchée. L'information spectrale est obtenue par un solveur de valeurs propres lors d'une phase préliminaire au calcul.

Dans une deuxième partie, on autorise une possible mise à jour entre chaque système. Une correction spectrale incrémentale est proposée. Des expérimentations numériques sont réalisées en utilisant la méthode GMRES-DR, d'une part parce qu'elle est efficace en tant que solveur linéaire, et d'autre part parce qu'elle permet une bonne approximation des petites valeurs propres au cours de la résolution linéaire. Des stratégies sont développées afin de sélectionner l'information spectrale la plus pertinente.

Ces approches ont été validées sur des problèmes de grande taille issus de simulations industrielles en électromagnétisme. Dans ce but, elles ont été implantées dans un code parallèle développé par EADS-CCR.

Mots-clés: systèmes linéaires denses et creux, méthodes de Krylov, GMRES, GMRES-DR, préconditionneur spectral incrémental, valeurs harmoniques de Ritz, simulations numériques de grande taille en électromagnétisme, calcul sientifique, calcul parallèle distribué.

Abstract

Many numerical simulations in scientific and engineering applications require the solution of a set of large linear systems involving the same coefficient matrix but different right-hand sides. Efficient methods for tackling this problem attempt to benefit from the previously solved right-hand sides for the solution of the next ones. This goal can be achieved either by recycling Krylov subspaces or by building preconditioner updates based on near invariant subspace information. In this thesis, we focus our attention on this last approach that attempts to improve a selected preconditioner.

In the first part, we consider only one update of the preconditioner for all the systems. This update consists of a spectral low-rank correction that shifts by one the smallest eigenvalues in magnitude of the matrix of the original preconditioned system. We perform experiments in the context of the GMRES method preconditioned by an approximate inverse preconditioner. The spectral information is computed by an eigensolver in a preprocessing phase.

In the second part, we consider an update of the preconditioner between each system. An incremental spectral correction of the preconditioner is proposed. We perform experiments using the GMRES-DR method, thanks to its efficiency as a linear solver and its ability to recover reliable approximations of the desired eigenpairs at run time. Suitable strategies are investigated for selecting reliable eigenpairs.

The efficiency of the proposed approaches is in particular assessed for the solution of large and challenging problems in electromagnetic applications. For this purpose, they have been implemented in a parallel industrial code developed by EADS-CCR.

Keywords: dense and sparse linear systems, Krylov methods, GMRES, GMRES-DR, incremental spectral preconditioner, harmonic Ritz value, large scale numerical simulations in electromagnetism, scientific computing, parallel distributed computing.

Remerciements

Je souhaiterais tout d'abord remercier Luc Giraud, mon directeur de thèse, chercheur sénior dans l'équipe ALGO, avec qui j'ai eu le plaisir de travailler. Je lui suis très reconnaissant pour son suivi, sa rigueur, ses conseils ainsi que sa disponibilité.

Je remercie Iain Duff, chef de l'équipe, pour toutes les questions soulevées pendant ma thèse et pour les nombreux commentaires apportés à mon travail.

Je remercie Stéphane Lanteri, directeur de recherche INRIA à Sophia-Antipolis et Angelika Bunse-Gerstner, professeur à l'université de Brême, qui m'ont fait l'honneur d'être rapporteurs de ma thèse.

Je remercie Abderrahmane Bendali, chef de l'équipe EMC, qui m'a fait l'honneur de présider mon jury.

Je remercie Serge Gratton, chercheur sénior de l'équipe ALGO, avec qui j'ai activement collaboré durant la deuxième moitié de ma thèse. J'ai pu profiter de ses nombreuses compétences et de la pertinence de ses remarques.

Je tiens particulièrement à remercier Julien Langou, ancien thésard de l'équipe, qui a été beaucoup présent lors du démarrage de cette thèse. Il m'a notamment permis de me familiariser avec le code industriel dont je me suis servi tout le long de ma thèse.

Je tiens également à remercier Bruno Carpentieri, autre ancien thésard de l'équipe, avec qui j'ai eu des sujets de recherche en commun et donc beaucoup d'échanges fructeux. Son aide, son soutien et sa bonne humeur m'ont été très précieux.

Je remercie toute l'équipe ALGO pour la bonne ambiance de travail, l'équipe CSG pour leur efficacité, Brigitte Yzel pour maints services rendus et pour sa bonne humeur communicative, et enfin le CERFACS pour son climat de convivialité.

Un grand merci également à mes amis qui m'ont soutenu durant cette aventure: Rodolphe, Elisabeth, Christof, Laurent, Mathieu, Sophie ...

Contents

Introduction

In this thesis we consider the iterative solution of sequences of linear systems with the same coefficient matrix but different right-hand sides. Such a situation occurs in many numerical simulations in scientific and engineering applications. Efficient solutions to tackle this problem can consist in studying new variants of iterative schemes, or in developing new preconditioning techniques or in combining the two approaches. In our work, we mainly focus our attention on the development and the study of new preconditioning variants. Our techniques apply to any situation where sequences of linear systems have to be solved. We assess their efficiency and robustness in particular on large challenging problems arising in boundary integral equations in electromagnetism. For this purpose, we have implemented these techniques in a large parallel distributed simulation code. An accurate numerical solution of these problems is required in the simulation of many industrial processes, such as the prediction of the Radar Cross Section (RCS) of arbitrarily shaped 3D objects like aircraft, the analysis of electromagnetic compatibility of electrical devices with their environment, and many others.

This manuscript is structured as follows. Because many of our numerical experiments are performed on linear systems arising from boundary integral equations in electromagnetism, we give in Chapter 1 an overview of the underlying physical models and of the discretization technique. We also provide a synthetic description of the Fast-Multipole method. The emergence of this latter technique introduced a significant technological gap because it enables the simulation of very large problems that were out of reach until then. For the solution of large scale dense problems, direct methods become impractical even on large parallel platforms because they require storage of N^2 single or 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. Consequently, iterative solvers are the only alternative and new solvers and preconditioners should be designed to efficiently solve these problems. In Chapter 2, we describe the variants of GMRES [76] that we consider for our numerical experiments. We focus on variants of GMRES because the GMRES method has been shown to be very robust for the solution of large electromagnetism problems [16, 20]. We conclude the first part of this manuscript with the description of candidate preconditioning techniques and illustrate their behaviour for the solution of one linear system.

In Part II, we study the numerical behaviour of a spectral two-level preconditioner for the calculation of a complete radar cross section on various real life test problems. The definition of this class of preconditioners [18] requires the knowledge of some eigen information. In this part, the spectral information is obtained in a preprocessing phase using an eigensolver. We illustrate the significant gains observed on large geometries on parallel distributed platforms.

In Part III, we investigate a new spectral preconditioning technique that relaxes the constraint of computing the eigen information in the preprocessing phase. In that approach, an approximation of the spectral information is extracted on the fly during the solution of one right-hand side and is used to solve the subsequent linear systems. We first study the numerical behaviour of this technique on small sparse linear systems and illustrate its efficiency on large problems in electromagnetism. Finally we give some conclusions and possible directions for future research.

I

Chapter 1

The electromagnetism problem

This chapter presents the main physical application we have considered throughout this manuscript to assess the performance of our numerical schemes. The first section describes the physical problem we focus on and the properties of the resulting matrices associated with the linear systems we intend to solve. The second section quickly describes the fast multipole method which implements a fast matrix-vector product and allows us to address the solution of very large problems. The last section presents the industrial code we use to test our numerical schemes in the solution of problems arising in electromagnetism. We also describe the test examples we consider in our numerical experiments.

1.1 Integral equations formulation and discretization

1.1.1 Problem definition

In Figure 1.1 we present a perfect conductor Ω lying in a vacuum. The outgoing normal on the boundary Γ of the object is denoted by $\vec{n}$. We denote by $\vec{E}$ the electric field and by $\vec{H}$ the magnetic field. An incident plane wave $(\vec{E}^{inc}, \vec{H}^{inc})$ with a frequency f and a pulsation $\omega = 2\pi f$ illuminates the object. We take as unknowns the generated scattered fields $(\vec{E}^{scat}, \vec{H}^{scat})$ in the outer domain Ω_{out}. They are solutions of the harmonic Maxwell equations:

$$\begin{cases} \vec{rot}\vec{E}^{scat} - i\omega\mu_0\vec{H}^{scat} = 0, & in\ \Omega_{out}, \\ \vec{rot}\vec{H}^{scat} + i\omega\epsilon_0\vec{E}^{scat} = 0, & in\ \Omega_{out}, \\ \vec{E}^{scat} \wedge \vec{n} = -\vec{E}^{inc} \wedge \vec{n}, & in\ \Gamma, \\ \lim\limits_{r\to+\infty} r \left| \sqrt{\epsilon_0}\vec{E}^{scat} - \sqrt{\mu_0}\vec{H}^{scat} \wedge \dfrac{\vec{r}}{r} \right| = 0, \end{cases} \tag{1.1}$$

where ϵ_0 is the permittivity of the vacuum and μ_0 the permeability of the vacuum. We denote by $Z_0 = \sqrt{\mu_0/\epsilon_0}$ the vacuum impedance. The total fields $(\vec{E}, \vec{H})$ are formed by:

$$\begin{cases} \vec{E} = \vec{E}^{inc} + \vec{E}^{scat}, \\ \vec{H} = \vec{H}^{inc} + \vec{H}^{scat}. \end{cases}$$

In order to solve the problem defined by Equation (1.1), we derive in the next section its integral formulation.

1.1.2 Integral equations

Let $\vec{\jmath}$ be the field tangent to the surface Γ and defined by: $\vec{\jmath} = \vec{n} \wedge \vec{H}$. The field $\vec{\jmath}$ is called the electric current and is the true unknown of the problem. The Stratton-Chu representation theorem

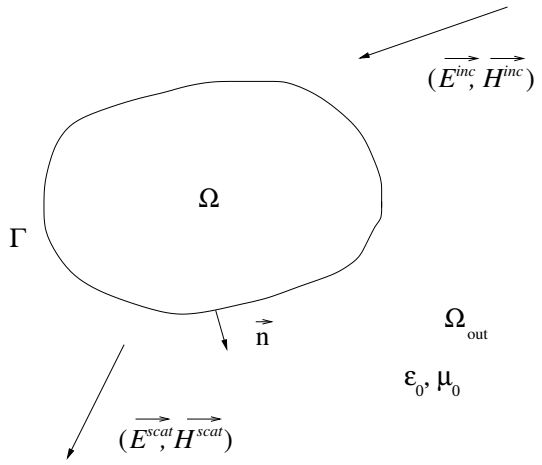


Figure 1.1: Scattered problem.

[25] gives the expression of the fields $\vec{E}$ and $\vec{H}$ at any point $y \notin \Gamma$ as a function of the field $\vec{\jmath}$:

$$\begin{cases} \vec{E}(y) = \vec{E}^{inc} + i\omega\mu_0 \displaystyle\int_\Gamma G(x,y)\vec{\jmath}(x)ds(x) + \frac{i}{w\epsilon_0} \int_\Gamma \vec{grad}_y G(x,y) div_\Gamma(\vec{\jmath}(x))ds(x), \\ \vec{H}(y) = \vec{H}^{inc} - \displaystyle\int_\Gamma \vec{grad}_y G(x,y) \wedge \vec{\jmath}(x)ds(x), \end{cases} \tag{1.2}$$

where:

$$G(x,y) = \frac{1}{4\pi|x-y|} e^{ik|x-y|}.$$

The Green's function $G(x,y)$ is the fundamental solution of the harmonic Helmholtz equation:

$$\Delta u + k^2 u = -\delta_0,$$

coupled with the Sommerfeld radiation condition:

$$\lim_{r\to+\infty} r\left(\frac{\partial u}{\partial r} - iku\right) = 0.$$

We define $k = \omega\sqrt{\epsilon_0\mu_0}$ and we denote by $div_\Gamma(\vec{\jmath})$ the surfacic divergence of $\vec{\jmath}$, defined on the plan tangent to the surface Γ. Using Equations (1.1) and (1.2) the electric equation becomes on Γ:

$$\frac{i}{kZ_0}\vec{E}^{inc}_T = \left[\int_\Gamma G(x,y)\vec{\jmath}(x)ds(x)\right]_T + \frac{1}{k^2}\left[\int_\Gamma \vec{grad}_y\left(G(x,y)\right) div_\Gamma(\vec{\jmath}(x))ds(x)\right]_T, \tag{1.3}$$

where $\vec{u}_T$ denotes the tangential component of the vector $\vec{u}$. This equation is called the Electric Field Integral Equation (EFIE). In a slightly more complicated way, an equation for the magnetic field can also be obtained and can be written as:

$$(\vec{n} \wedge \vec{H}^{inc})(y) = \frac{1}{2}\vec{\jmath}(y) + \vec{n}(y) \wedge \int_\Gamma \left(\vec{grad}_x(G(x-y)) \wedge \vec{\jmath}(x)\right) ds(x). \tag{1.4}$$

Equation (1.4) is called the Magnetic Field Integral Equation (MFIE). The problem of inner resonant frequencies leads the EFIE and MFIE equations to have non-unique solutions for certain

values of k. In order to prevent such a problem, we can introduce the Combined Field Integral Equation (CFIE), which is a linear combination of EFIE defined by (1.3) and MFIE defined by (1.4): $CFIE = \alpha EFIE + (1-\alpha)\frac{i}{k}MFIE$. The constant α is defined with $0 < \alpha < 1$. In pratice, $\alpha = 0.2$ is often used.

1.1.3 Discretization

Let $\vec{\jmath}^{\,'}$ be a test field, tangent to the surface Γ. By multiplying Equations (1.3) and (1.4) by $\vec{\jmath}^{\,'}$, and integrating them on the surface Γ, we obtain the variational formulations for the fields $\vec{E}$ and $\vec{H}$.

In variational form, the EFIE formulation becomes:

Find the current $\vec{\jmath}(x)$ *such as, for any current test function* $\vec{\jmath}^{\,'}(y)$*, we have:*

$$\begin{cases} \displaystyle\int_\Gamma\int_\Gamma G(x,y)\left(\vec{\jmath}(x)\vec{\jmath}^{\,'}(y) - \frac{1}{k^2}div_\Gamma(\vec{\jmath}(x))div_\Gamma(\vec{\jmath}^{\,'}(y))\right)ds(x)ds(y) \\ \qquad\qquad = \dfrac{i}{kZ_0}\displaystyle\int_\Gamma \vec{E}^{inc}(y)\vec{\jmath}^{\,'}(y)ds(y). \end{cases} \tag{1.5}$$

In variational form, the MFIE formulation becomes:

Find the current $\vec{\jmath}(x)$ *such as, for any current test function* $\vec{\jmath}^{\,'}(y)$*, we have:*

$$\begin{cases} \dfrac{1}{2}\displaystyle\int_\Gamma \vec{\jmath}(y)\vec{\jmath}^{\,'}(y)dy + \int_\Gamma \vec{\jmath}^{\,'}(y)\vec{n}(y)\wedge\left(\int_\Gamma \vec{grad}_x(G(x,y))\wedge\vec{\jmath}(x)ds(x)ds(y)\right) \\ \qquad\qquad = \displaystyle\int_\Gamma \vec{\jmath}^{\,'}(y)\vec{n}(y)\wedge\vec{H}^{inc}(y)dy. \end{cases} \tag{1.6}$$

We suppose that the surface of Γ is discretized with triangles. We use the standard discretization of Raviart-Thomas [69], where the degrees of freedom are associated with the edges. The basis functions $\vec{\Phi}_p$ are associated with the edges of the surfacic mesh; its value is zero everywhere except on the two triangles T^+ and T^- that share the edge ed_p. On these two triangles, $\vec{\Phi}_p$ is defined by :

$$\vec{\Phi}_p(x) = \begin{cases} \dfrac{1}{2.area(T^+)}\left(x - V^+\right), \\ -\dfrac{1}{2.area(T^-)}\left(x - V^-\right), \end{cases}$$

where V^+ is the vertex opposite the edge ed_p in the triangle T^+ and V^- the vertex opposite the edge ed_p in the triangle T^-. The field $\vec{\jmath}$ can be written in terms of basis functions as:

$$\vec{\jmath}(x) = \sum_{q=1}^{m} j_q\vec{\Phi}_q(x)$$

where m is the total number of edges. The components j_q can be interpreted as the flux of the current $\vec{\jmath}(x)$ across the edge ed_q.

By taking $\vec{\jmath}^{\,'} = \vec{\Phi}_p$ with $1 \le p \le m$, we obtain a linear system of order m where the unknowns are the components j_q. The m linear equations are:

$$\sum_{q=1}^{m}\mathbb{A}_{p,q}j_q = f_p.$$

In matrix form, this gives:

$$\mathbb{A}.j = f, \tag{1.7}$$

For the EFIE formulation, the entry (p,q) of $\mathbb{A}$ is:

$$\mathbb{A}_{p,q} = \int_\Gamma \int_\Gamma G(x,y) \left(\vec{\Phi}_p(x)\vec{\Phi}_q(y) - \frac{1}{k^2} div_\Gamma(\vec{\Phi}_p(x)) div_\Gamma(\vec{\Phi}_q(y)) \right) ds(x)ds(y),$$

and the entry p of f is

$$f_p = \frac{i}{kZ_0} \int_\Gamma \vec{E}^{inc}(y)\vec{\Phi}_p(y)ds(y).$$

Consequently, the coefficient matrix $\mathbb{A}$ is symmetric non-Hermitian.

For the MFIE formulation, the entry (p,q) of $\mathbb{A}$ is:

$$\mathbb{A}_{p,q} = \frac{1}{2} \int_\Gamma \vec{\Phi}_q(y)\vec{\Phi}_p ds(y) + \int_\Gamma \int_\Gamma \vec{grad}_x(G(x,y)) \wedge \vec{\Phi}_q(x) \cdot \left(\vec{\Phi}_p(y) \wedge \vec{n}(y) \right) ds(x)ds(y),$$

and the entry p of f is

$$f_p = \int_\Gamma \vec{n}(y) \wedge \vec{H}^{inc}(y) \cdot \vec{\Phi}_p(y)ds(y).$$

Consequently, the coefficient matrix $\mathbb{A}$ is no longer symmetric.

As the CFIE formulation leads to linear systems that are easy to solve, it is preferably used by the electromagnetics community. When the scattering object presents a cavity whose the length of the outline is very small, or when the mesh shows some salient edges, the outgoing normal can no longer be defined. The MFIE and CFIE formulations are no longer applicable as they require the definition of the outgoing normal $\vec{n}$. The EFIE formation becomes the only alternative to deal with such cases. Unfortunately it gives rise to linear systems that are more difficult to solve.

Dealing with objects of large size in terms of wavelength, leads to dense matrices of very large size. For example, the surfacic mesh of a sphere of diameter $150\,\lambda$ leads to a dense linear system with 25,579,200 unknowns [85]. The resulting dense matrix can no longer be built. The fast multipole method briefly described in the next section becomes the only alternative to perform matrix-vector products for this class of huge problems.

1.2 The parallel fast multipole method

1.2.1 Presentation of the method

The Fast Multipole Method (FMM), introduced by Greengard and Rokhlin in [44], provides an algorithm for computing approximate matrix-vector products for electromagnetic scattering problems. The method is fast in the sense that the computation of one matrix-vector product costs $\mathcal{O}(n \log n)$ arithmetic operations instead of the usual $\mathcal{O}(n^2)$ operations for a regular dense matrix-vector product. The storage is also reduced from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$. Owing to these desirable properties its use in combination with iterative solvers is the only alternative for the solution of large problems.

The basic idea of the algorithm is to compute interactions amongst degrees of freedom in the mesh at different levels of accuracy depending on their physical distance. Single and multilevel variants of the FMM exist. In the one-level FMM, the 3D obstacle is entirely enclosed in a large domain, and the domain is divided into eight boxes (four in 2D). Each box is recursively divided until the size of the boxes is small enough compared with the wavelength. The near-field interactions, that is those between degrees of freedom from neighbouring boxes, are computed from Equation (1.3) using the regular expression of the discrete Green's function. The neighbourhood

of a box is defined by the box itself and its 26 adjacent neighbours (eight in 2D). The far-field contribution from far away cubes is computed approximately. More precisely, for each far away box, multipole coefficients are computed from Equation (1.3) using a truncated series expansion of the Green's function

$$G(x,y) = \sum_{p=1}^{P} \psi_p(x)\phi_p(y), \tag{1.8}$$

which separates the Green's function into two sets of terms, ψ_i and ϕ_j, that depend on the observation point x and the source point y, respectively. In Equation (1.8) the integer P is generally very small, and the origin of the expansion is near the source point y while the observation point x is far away. Multipole coefficients of far-away boxes are summed together to compute local coefficients for the observation cube, and the total effect of the far field on each observation point is evaluated from the local coefficients (see Figure 1.2 for a 2D illustration). Local and multipole coefficients can be computed in a preprocessing step, reducing the overall computational cost of the matrix-vector product to $\mathcal{O}(n^{3/2})$ in the basic one-level algorithm.

In the hierarchical multilevel algorithm, the box-wise partitioning of the obstacle is carried out until the size of the smallest box is generally half a wavelength, and a tree-structured data is used at each level. In particular, only non-empty cubes are indexed and recorded in the data structure. The resulting tree is called the *oct-tree* (see Figure 1.3) and its leaves are generally referred to as the *leaf-boxes*. The oct-tree provides a hierarchical representation of the computational domain partitioned by boxes: each box has one parent in the oct-tree, except for the largest cube which encloses the whole domain, and up to eight children. Obviously, the leaf-boxes have no children. Multipole coefficients are computed for all cubes starting from the lowest level of the oct-tree, that is from the leaf-boxes, and then recursively for each parent cube by summing together multipole coefficients of their children. An observation cube is any cube which can have children. For each observation cube, an interaction list is defined which consists of those cubes that are not neighbours of the cube itself but whose parent is a neighbour of the cube's parent (see Figure 1.4 for a 2D representation, dashed lines). The interactions of degrees of freedom within neighbouring boxes are computed exactly (see Figure 1.4, the gray boxes), while the interactions between cubes that are in the interaction list are computed using the FMM. All the other interactions are computed hierarchically on a coarser level traversing the oct-tree. Both the computational cost and the memory requirement of the algorithm are of order $\mathcal{O}(n \log n)$. Further information on the algorithmic steps and recent theoretical investigations of the FMM can be found in [26, 27], and details on the parallel implementation issues are given in [41, 43, 85, 90].

1.2.2 Different levels of accuracy

One feature of the FMM is that it can be tuned to be more or less accurate and fast. By adjusting parameters such as the size of the octree's leaves or the order of the expansion, the method may be either very accurate (and not so fast) or less accurate (and really fast). In our work we consider two accuracy levels in order to exploit this feature in iterative solvers:

- Accurate FMM: the FMM is very accurate. It governs the final accuracy of the solution. We refer to the Section 2.5.1 for related considerations.
- Fast FMM: the FMM is very fast while keeping the forward error below a few percents. This feature will be used to implement efficient inner-outer scheme. We refer to the Section 2.3 for the description of linear solvers that can exploit this feature.

A complete description of this feature is given in [85]. The next section presents two classic calculations in scattering problems. The monostatic calculation studies the sensitivity of an incident electromagnetic wave on the scattering of an object. The bistatic calculation estimates the electromagnetic field generated by a source. We finish with a description of the geometries and the test code we use in our numerical experiments.

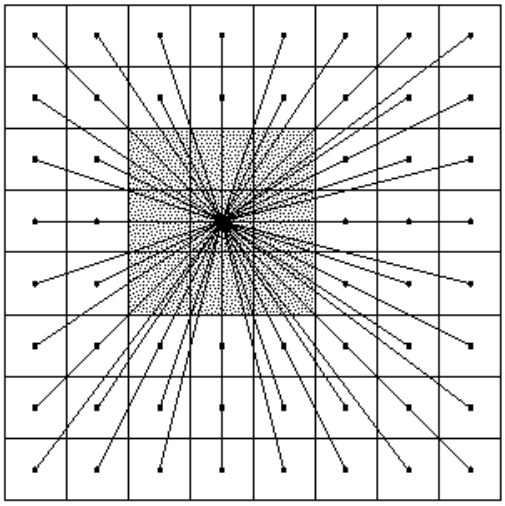


Figure 1.2: Interactions in the one-level FMM. For each leaf-box, the interactions with the gray neighbouring leaf-boxes are computed directly. The contribution of far away cubes are computed approximately. The multipole expansions of far away boxes are translated to local expansions for the leaf-box; these contributions are summed together and the total field induced by far away cubes is evaluated from local expansions.

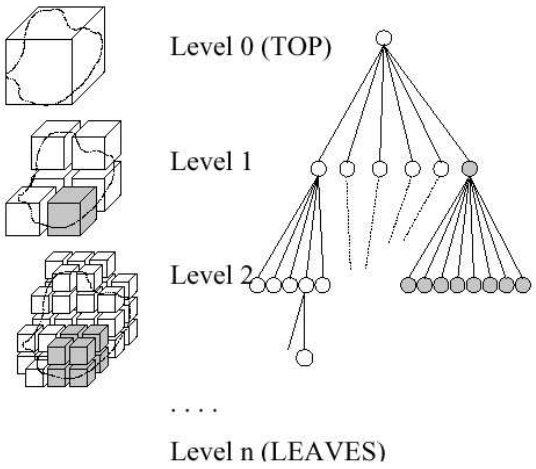


Figure 1.3: The oct-tree in the FMM algorithm. The maximum number of children is eight. The actual number corresponds to the subset of eight that intersect the object (courtesy of G. Sylvand, EADS-CCR).

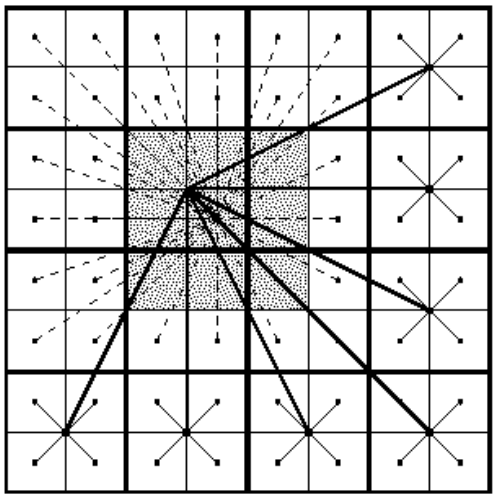


Figure 1.4: Interactions in the multilevel FMM. The interactions for the gray boxes are computed directly. We denote by dashed lines the interaction list for the observation box, that consists of those cubes that are not neighbours of the cube itself but whose parent is a neighbour of the cube's parent. The interactions of the cubes in the list are computed using the FMM. All the other interactions are computed hierarchically on a coarser level, denoted by solid lines.

1.3 The radar cross section calculation

1.3.1 Monostatic and bistatic calculation

The Figure 1.5 shows the situation of a point P. For an origin 0, such a point can be located in the Cartesian basis $(\hat{e}_a, \hat{e}_b, \hat{e}_c)$ by (a, b, c) or in the spherical basis $(\hat{u}_r, \hat{u}_\theta, \hat{u}_\varphi)$ by (r, θ, φ).

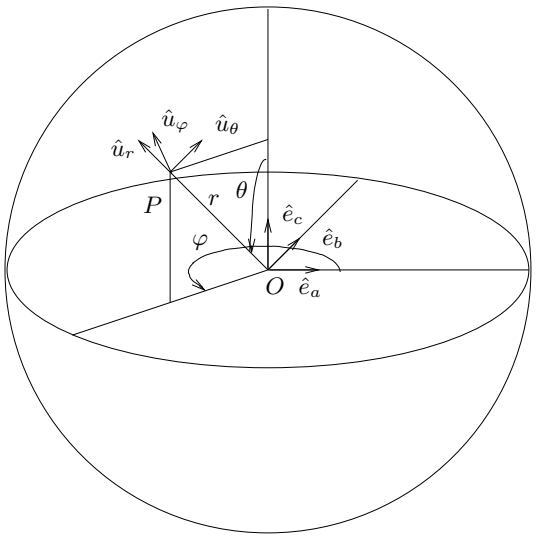


Figure 1.5: spherical coordinates.

The wave vector $\vec{k}$ is oriented from P to 0, i.e. $\vec{k} = -k\hat{u}_r$. As the incident wave is planar, we have $\vec{E}^{inc} \wedge \hat{u}_r = 0$, that is $\vec{E}^{inc}$ is defined in the plan spanned by $< \hat{u}_\theta, \hat{u}_\varphi >$. The expression for the incident plane wave is then:

$$\vec{E}^{inc}(\vec{r}, \theta, \varphi) = (p_\theta \hat{u}_\theta) e^{i\vec{k}\cdot\vec{r}} + (p_\varphi \hat{u}_\varphi) e^{i\vec{k}\cdot\vec{r}}, \tag{1.9}$$

where p_θ and p_φ are two complex numbers. We call $(p_\theta, p_\varphi) = (1, 0)$ the φ polarization, and $(p_\theta, p_\varphi) = (0, 1)$ the θ polarization.

The monostatic calculation consists in sending an incident wave $\vec{E}^{inc}$ in a direction (r, θ, φ) and recovering the energy of the back-scattered field in the direction $(-r, \theta, \varphi)$. For an angular section corresponding to a variation of θ (resp. ϕ) for a fixed value of ϕ (resp. θ), a monostatic computation is performed for each incident angle (θ_ℓ, φ) (resp. (θ, φ_ℓ)). This procedure is called the monostatic Radar Cross Section calculation. It implies that the number of linear systems to be solved will be equal to the number of incident waves considered. The coefficient matrix remains the same for all the waves, only the right-hand sides change. For an angle (θ_ℓ, φ), the associated linear system can be written:

$$\mathbb{A}.j_{\theta_l} = f_{\theta_l}.$$

Once this system has been solved, we obtain the quantity:

$$\vec{\jmath}_{\theta_l}(x) = \sum_{q=1}^{m} j_q \vec{\Phi}_q(x).$$

Definition 1.3.1 *The monostatic Radar Cross Section (RCS)* [24] *in the direction* (θ, φ) *is the ratio of the energies between the scattered field* $\vec{E}^{scat}$ *at infinity and the incident field* $\vec{E}^{inc}$,

$$RCS = 10 \times log_{10}(\gamma), \tag{1.10}$$

where γ *is defined as:*

$$\gamma = \lim_{r \to +\infty} 4\pi r^2 \frac{\left|\vec{E}^{scat}(\vec{r})\right|^2}{\left|\vec{E}^{inc}\right|^2}. \tag{1.11}$$

The unit of measurement is the decibel by square meter $(db \times m^2)$.

The quantity γ can be written as an expression of the far field:

$$a_\infty(\theta_l) = \frac{ikZ_0}{4\pi} \int_\Gamma (\hat{u}_r(\theta_l) \wedge (\vec{\jmath}_{\theta_l}(x) \wedge \hat{u}_r(\theta_l)))\, ds(x). \tag{1.12}$$

Then, the back-scattered radar cross section for an incident angle (θ_l, φ) is computed as:

$$\mathrm{RCS}(\theta_l) = 20 \times \log_{10}(|a_\infty(\theta_l)|).$$

In Figure 1.6, we plot the computed RCS for a Cobra discretized with 60 695 degrees of freedom (we refer to Figure 1.7 for the shape of this geometry), when varying θ for a fixed value of φ (φ polarization). The point P on the curve has the following meaning. An incident plane wave $(\theta_l, \varphi) = (40^o, 0^o)$ is illuminating the Cobra. We recover the energy of the associated back-scattered field from the same position (θ_l, φ). Depicted in a decibel scale in the curve, the value of the energy $\mathrm{RCS}(\theta_l = 40^o)$ is equal to -5. By varying θ_l from 0 to 90^o we obtain the whole curve. One application of the radar cross section is in radar furtivity. In that context, the aim is to obtain a RCS as small as possible for certain angular sections of the object of interest. It will prevent any conclusions about the original shape of the object on these angular sections.

The bistatic calculation consists in sending only one incident wave $\vec{E}^{inc}$ in a direction $(\vec{r}, \theta_l, \varphi)$ and recovering the energy of the back-scattered field in any observation direction $(-\vec{r}, \theta_h, \varphi)$. For

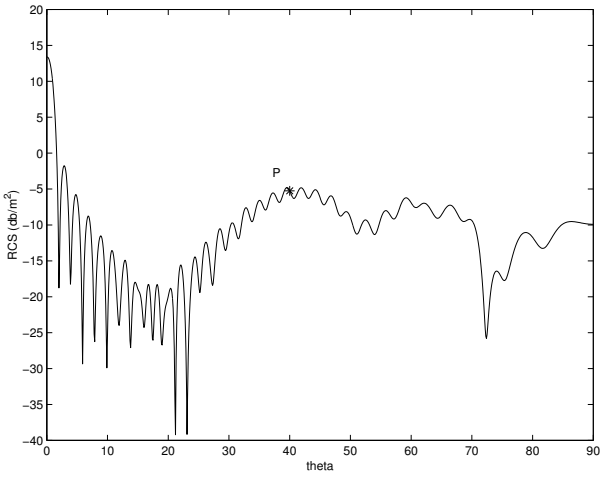


Figure 1.6: monostatic Radar Cross Section (RCS) for the Cobra 60695.

this case, only one incident wave is required (only one system has to be solved). The expression of the far field for the bistatic calculation is given by:

$$a_\infty(\theta_l, \theta_h) = \frac{ikZ_0}{4\pi} \int_\Gamma (\hat{u}_r(\theta_h) \wedge (\vec{\mathrm{J}}_{\theta_l}(x) \wedge \hat{u}_r(\theta_h)))\, ds(x).$$

One application of the bistatic calculation is the data transmission between a scattering antenna on a satellite in space and a receiver on earth. The maxima on the bistatic RCS curve give the best scattering directions of the antenna. Such directions are used to fix inclination angles of the receiver.

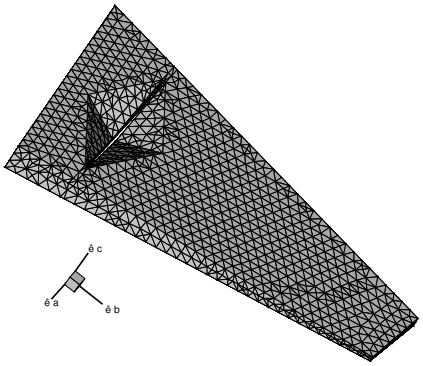
In this manuscript, we focus our attention on the monostatic case using both θ polarization and φ polarization.

1.3.2 Test geometries

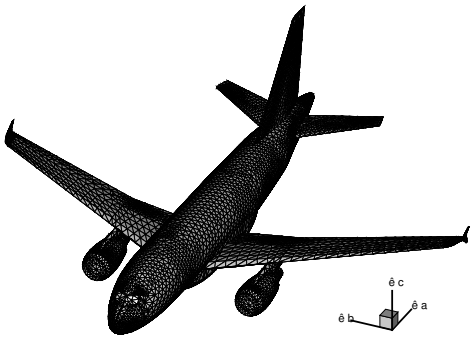
As scatterer object, we choose the four geometries depicted in Figure 1.7. They consist of a wing with a hole referred to as Cetaf, an aircraft from a European manufacturer, an air intake referred to as Cobra, and finally an Almond. The Cetaf and Almond cases are classic test problems in the computational electromagnetics community; the other two have been kindly provided to us by EADS–CCR.

1.3.3 The simulation code and the targeted parallel computers

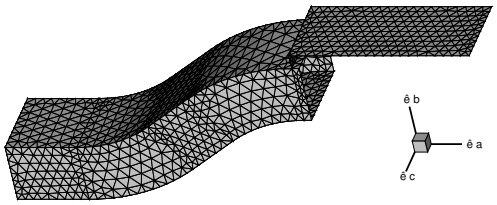
This section is devoted to a brief presentation of the industrial code developed by EADS-CCR, that we use for our numerical investigations. This code, called AS_ELFIP, simulates scattering problems in electromagnetic calculations. It is a joint effort of three partners. The main contributor is EADS-CCR: they develop such a software for solving Maxwell and Helmholtz equations in the frequency domain based on a boundary finite-element discretization. The linear systems can be solved either by direct or iterative methods. The second contributor, Guillaume Sylvand, has developed during his PhD thesis [85] at CERMIC/INRIA, an implementation of the fast multipole method in this code, for sequential and parallel distributed platforms. And the last contribution to this work comes from the Parallel Algorithm Project at CERFACS. A PhD thesis by Bruno Carpentieri [16] shows the performance of an efficient Frobenius norm minimization preconditioner



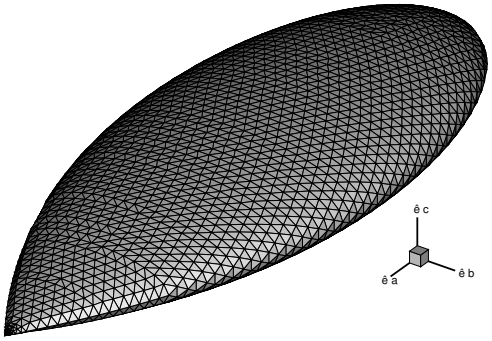
(a) Cetaf



(b) Aircraft



(c) Cobra



(d) Almond

Figure 1.7: Various geometries used in the numerical experiments.

in the electromagnetic context. A second PhD thesis from Julien Langou [53] deals with numerical techniques for the solution of linear systems with multiple right-hand sides. All these recent developments have been integrated in the AS_ELFIP code. From a technical point of view, the code is written mainly in the C language with several Fortran 77 libraries. It is parallel and suited for shared or distributed memory computers. Moreover the code uses out-of-core data structures to reduce memory requirements so enabling the possibility of performing huge calculations. In particular all the vectors involved in the linear algebra operations are out-of-core; for a detailed description of the other features we refer the reader to [39, 85].

Finally we mention that all the numerical experiments reported in this manuscript have been performed on a Compaq Alpha Server which is a cluster of Symmetric Multi-Processors. A server located at CERFACS offers ten nodes of four DEC Alpha processors EV 68 each. Each processor has a frequency of 1GHz for a theoretical peak of 2 GFlops. Each node has 8 Gb of memory and a local temporary disk of 7 Gb. A second server located at CEA (Commissariat à l'Énergie Atomique) offers ninety-five nodes consisting of four DEC Alpha processors EV 68. Each processor has a frequency of 1.25 GHz for a theoretical peak of 2.5 GFlops. Each node has 4 Gb of memory and a local temporary disk of 12 Gb.

Chapter 2

Some variants of the GMRES linear solver

This chapter describes several iterative linear solvers suitable for the solution of large non-Hermitian linear systems. All these methods are designed to solve a linear system of the form:

$$Ax = b, \tag{2.1}$$

where $A \in \mathbb{C}^{n\times n}$ is a $n \times n$ nonsingular matrix, $x \in \mathbb{C}^n$ and $b \in \mathbb{C}^n$. In many cases, such methods converge slowly, or even diverge. The convergence of iterative methods may be improved by transforming the system (2.1) into another system which is easier to solve. A preconditioner is a matrix that realizes such a transformation. If M is a nonsingular matrix which approximates A^{-1}, then the transformed linear system:

$$MAx = Mb, \tag{2.2}$$

might be solved faster. The system (2.2) is preconditioned from the left, but one can also precondition from the right:

$$AMt = b. \tag{2.3}$$

Once the solution t is obtained, the solution of the system (2.1) is recovered by $x = Mt$.

The Section 2.1 briefly describes the iterative procedure and the GMRES algorithm [76]. We give in Section 2.2 a presentation of the GMRES-DR [64] method, a solver that can recover eigen information of the preconditioned matrix during the iterative solve. In particular, we show that an estimation of the accuracy of the computed eigen information can be computed using by-products of the algorithm. Section 2.3 is devoted to the Flexible GMRES [74] method which enables us to vary the preconditioner at each iteration. Finally, an approach for multiple right-hand sides is given in Section 2.4, that corrects the initial guesses of subsequent systems using approximations of the solution of the current solve. We conclude in Section 2.5 by indicating some features shared by all these solvers.

2.1 The GMRES method

The Generalized Minimum RESidual (GMRES) method was proposed by Saad and Schultz in 1986 [76] for the solution of large non-Hermitian linear systems. GMRES belongs to the class of Krylov based iterative methods. For the sake of generality, we describe this method for linear systems whose entries are complex; everything also extends to real arithmetic.

Let $x_0 \in \mathbb{C}^n$ be an initial guess for the solution to the linear system (2.1) and $r_0 = b - Ax_0$ be its corresponding residual. At step j, the GMRES algorithm builds an approximation of the

solution to (2.1) of the form

$$x_j = x_0 + Q_j y_j, \tag{2.4}$$

where $y_j \in \mathbb{C}^j$ and $Q_j = [q_1, \cdots, q_j]$ is an orthonormal basis for the Krylov space of dimension j defined by

$$\mathcal{K}(A, r_0, j) = \text{span}\left\{r_0, Ar_0, \ldots, A^{j-1}r_0\right\}.$$

The vector y_j is determined so that the 2–norm of the residual $r_j = b - Ax_j$ is minimized over $x_0 + \mathcal{K}(A, r_0, j)$. The basis Q_j for the Krylov subspace $\mathcal{K}(A, r_0, j)$ is obtained via the well known Arnoldi process [6]. The orthogonal projection of A onto $\mathcal{K}(A, r_0, j)$ results in an upper Hessenberg matrix $H_j = Q_j^H A Q_j$ of order j. The Arnoldi process satisfies the relationship

$$AQ_j = Q_j H_j + h_{j+1,j} q_{j+1} e_j^T, \tag{2.5}$$

where e_j is the j^{th} canonical basis vector. Equation (2.5) can be rewritten in matrix form as

$$AQ_j = Q_{j+1}\bar{H}_j,$$

where

$$\bar{H}_j = \begin{bmatrix} H_j \\ 0 \cdots 0 \; h_{j+1,j} \end{bmatrix}$$

is an $(j+1) \times j$ matrix.

Let $q_1 = r_0/\beta$ where $\beta = \|r_0\|_2$. The residual r_j associated with the approximate solution x_j defined by (2.4) satisfies

$$\begin{aligned} r_j &= b - Ax_j = b - A(x_0 + Q_j y_j) \\ &= r_0 - AQ_j y_j = r_0 - Q_{j+1}\bar{H}_j y_j \\ &= \beta q_1 - Q_{j+1}\bar{H}_j y_j \\ &= Q_{j+1}(\beta e_1 - \bar{H}_j y_j). \end{aligned}$$

Because Q_{j+1} is a matrix with orthonormal columns, the residual norm $\|r_j\|_2 = \|\beta e_1 - \bar{H}_j y_j\|_2$ is minimized when y_j solves the linear least-squares problem

$$\min_{y \in \mathbb{C}^j} \|\beta e_1 - \bar{H}_j y\|_2. \tag{2.6}$$

We denote by y_j the solution of (2.6). Therefore, $x_j = x_0 + Q_j y_j$ is an approximate solution of (2.1) for which the residual is minimized over $x_0 + \mathcal{K}(A, r_0, j)$. The GMRES method owes its name to this minimization property that is its key feature as it ensures the decrease of the residual norm associated with the sequence of iterates.

In exact arithmetic, GMRES converges in at most n steps. However, in practice, n can be very large and the storage of the orthonormal basis Q_j may become prohibitive. On top of that, the orthogonalization of q_j with respect to the previous vectors $q_1, \cdots, q_{j-1}$ requires $4nj$ flops, for large j, the computational cost of the orthogonalization scheme may become very expensive. The restarted GMRES method is designed to cope with these two drawbacks. Given a fixed m, the restarted GMRES method computes a sequence of approximate solutions x_j until x_j is acceptable or $j = m$. If the solution was not found, then a new starting vector is chosen on which GMRES is applied again. Often, GMRES is restarted from the last computed approximation, i.e. $x_0 = x_m$ to comply with the monotonicity property of the norm decrease even when restarting. The process is iterated until a good enough approximation is found. We denote by GMRES(m) the restarted GMRES algorithm for a projection size of at most m. A detailed description of the restarted GMRES with right preconditioner and modified Gram-Schmidt algorithm as orthogonalization scheme is given in Algorithm 1.

An implementation of the GMRES algorithm for real and complex, single and double precision arithmetic suitable for serial, shared memory and distributed memory computers is available from the Web at the following URL:

http://www.cerfacs.fr/algor/Softs/

The implementation is based on the reverse communication mechanism for the matrix-vector product, the preconditioning and the dot product computations. The AS_ELFIP code uses an out-of-core version of this package.

Algorithm 1 Right preconditioned restarted GMRES algorithm: GMRES(m).

Choose a convergence threshold ε ;
Choose an initial guess t_0 ;
$r_0 = b - AMt_0$;
for $s = 1, \ldots$ **do**
  $\beta = \|r_0\|$;
  $q_1 = r_0/\beta$;
  for $j = 1, \ldots, m$ **do**
    $w = AMq_j$;
    for $i = 1$ `to` j **do**
      $h_{i,j} = q_i^H w$;
      $w = w - h_{i,j} q_i$;
    end for
    $h_{j+1,j} = \|w\|$;
    $q_{j+1} = w/h_{j+1,j}$;
    Find y_j the least-squares solution of $\min_y \|\beta e_1 - \bar{H}_j y\|$
    if (stopping criterion $\leq \varepsilon$) **then**
      $t = t_0 + Q_j y_j$;
      goto 24 ;
    end if
  end for
  $t_0 = t_0 + Q_m y_m$;
  $r_0 = b - AMt_0$;
end for
$x = Mt$;

2.2 The GMRES-DR method

It is well known that the convergence of Krylov methods for solving the linear system often depends to a large extent on the eigenvalue distribution. It is often observed that removing or deflating the small eigenvalues can greatly improve the convergence rate. One way to deflate unwanted eigenvalues is to add the corresponding eigenvectors to the Krylov subspace [73]. This is the philosophy that governed the development of the GMRES-DR method [64]. This method recycles some approximate eigenvectors from one restart to the next by augmenting the next Krylov subspace with these approximate eigenvectors. The purpose of this section is to describe this algorithm and to illustrate some of its properties. We first present some procedures for extracting approximate eigenpairs from a subspace for large non-Hermitian matrices. We then describe the GMRES-DR algorithm which is a way to combine such eigen information retrieval with a restarted Arnoldi method. The last part is devoted to a criterion to estimate the accuracy of the eigenvectors com-

puted by the GMRES-DR algorithm. This criterion is estimated using by-products of the method, i.e. at very low extra cost.

2.2.1 The Rayleigh-Ritz procedures

For any subspace $S \subseteq \mathbb{C}^n$, a vector $y \in S$ is a Ritz vector of the matrix A with Ritz value λ if

$$w^H(Ay - \lambda y) = 0, \quad \forall\ w \in S. \tag{2.7}$$

Let $Q \in \mathbb{C}^{n\times m}$ be an orthonormal matrix whose columns span S. Let $z_1 \in \mathbb{C}^m$ and $z_2 \in \mathbb{C}^m$ be so that $y = Qz_1$ and $w = Qz_2$. Equation (2.7) can be written:

$$z_2^H(Q^HAQz_1 - \lambda z_1) = 0, \quad \forall z_2 \in \mathbb{C}^m, \tag{2.8}$$

which leads to the standard eigenvalue problem:

$$Q^HAQg = \lambda g,$$

where z_1 is replaced by g in Equation (2.8). Let (λ_i, g_i) be the eigenpairs of Q^HAQ. The λ_i's are approximate eigenvalues of A and are referred to as Ritz values. The Qg_i's are approximate eigenvectors of A and are referred to as Ritz vectors. This approach is described in [62, 72] and is called the regular Rayleigh-Ritz procedure. Frequently, we choose $S = \mathcal{K}(A, r_0, m)$ for which an orthonormal basis is built using the Arnoldi procedure. We then have $S = span\{q_1, \cdots, q_m\}$, with $Q_m = [q_1, \cdots, q_m]$ such that $Q_m^HAQ_m = H_m$, where H_m is an upper Hessenberg matrix. In that case, computing the Ritz eigenpairs reduces to solving the standard eigenvalue problem

$$Hg = \lambda g.$$

Unfortunately, the Rayleigh-Ritz procedure usually does a better job in finding estimates for exterior eigenvalues than for interior eigenvalues. A variant of the Rayleigh-Ritz procedure has been proposed in [62] for finding interior eigenvalues and is described below. It defines the harmonic Ritz values as the Ritz values of A^{-1} with respect to the space AS; that is

$$w^H(A^{-1}y - \alpha y) = 0, \quad \forall\ w \in AS, \tag{2.9}$$

where $\alpha \in \mathbb{C}$. Let $Q \in \mathbb{C}^{n\times m}$ be an orthonormal matrix whose columns span S. Equation (2.9) with $y = AQz_1$ and $w = AQz_2$ gives us:

$$z_2^HQ^HA^H(A^{-1}AQz_1 - \alpha AQz_1) = z_2^HQ^HA^H(Qz_1 - \alpha AQz_1) = 0, \quad \forall z_2 \in \mathbb{C}^m$$

which leads to the generalized eigenvalue problem:

$$Q^HA^HQg = \alpha Q^HA^HAQg, \tag{2.10}$$

where z_1 is replaced by g. Let us choose $S = \mathcal{K}(A, r_0, m)$ for which an orthonormal basis is built using the Arnoldi procedure. We then have $S = span\{q_1, \cdots, q_m\}$, with $Q_m = [q_1, \cdots, q_m]$ such that $Q_m^HAQ_m = H_m$ and $AQ_m = Q_{m+1}\bar{H}_m$. Because $\bar{H}_m^H\bar{H}_m = H_m^HH_m + h_{m+1,m}^2e_me_m^T$, we have $H_m^Hg = \alpha(H_m^HH_m + h_{m+1,m}^2e_me_m^T)g$; the generalized eigenvalue problem (2.10) reduces to

$$(H_m + h_{m+1,m}^2fe_m^T)g = \alpha^{-1}g, \tag{2.11}$$

where $f = H_m^{-H}e_m$. The value $\theta = \alpha^{-1}$ is the harmonic Ritz value and the corresponding harmonic Ritz vector is Q_mg. Because the Ritz values α are supposed to well approximate the large eigenvalues of A^{-1}, the values θ are expected to be good approximations of the eigenvalues of A close to zero.

From a computational point of view, the difference between the Ritz eigenpairs and the harmonic Ritz eigenpairs is very small. The former requires the computation of the eigenpairs of H_m and the latter are obtained as the eigenpairs of $H_m + h_{m+1,m}^2fe_m^T$. Forming this last quantity requires $\mathcal{O}(m)$ operations and is negligible compared to the size of the matrix A.

2.2.2 Description of the GMRES-DR algorithm

The first restart of the GMRES-DR algorithm is a regular restart of the standard GMRES(m) algorithm where we denote by r the residual vector at the end of the first restart. At the end of the first restart, k harmonic eigenpairs (θ_i, g_i) associated with the k harmonic Ritz values with smallest modulus are extracted among the m available pairs. For the second restart, the first k columns of Q are formed by orthonormalizing the k harmonic eigenvectors g_i. Then r is orthogonalized against them to form q_{k+1}. Let H be the Hessenberg matrix from the Arnoldi recurrence. The full leading $k+1$ by $k+1$ portion of H is formed following Step 5 of Algorithm 2. From there, the rest of H and Q can be generated with the usual Arnoldi approach. This procedure generates the space $\mathcal{K}$:

$$\mathcal{K} = \text{span}\left\{g_1, g_2, ..., g_k, r, Ar, A^2r, \ldots, A^{m-k-1}r\right\}.$$

In [64], it is proved that the space $\mathcal{K}$ is a Krylov space. Distinct harmonic Ritz values for each restart are considered. If the matrix is real, conjugate harmonic eigenvectors can appear. If all the complex eigenvectors have their conjugates among the k selected eigenvectors, for each conjugate pair a real basis of the complex subspace spanned by this pair is formed. This real basis will be recycled for the next Krylov subspace. If a complex eigenvector does not have its conjugate, it is discarded. In this way, we prevent the next Krylov subspace from becoming complex.

The storage required by the GMRES-DR(m, k) is the same as for GMRES(m). Indeed, the first restart of GMRES-DR creates a subspace of size m. All the other restarts start with k eigenvectors in their Krylov subspaces and stop when the size of subspace is equal to m. The GMRES-DR algorithm performs m matrix-vector products in the first restart and $(m-k)$ matrix-vector products in the others.

2.2.3 Some remarks on the approximation of the eigenvectors

Because we stop the GMRES-DR iterations when the stopping criterion for the linear solver is less than a prescribed tolerance ε, there is no guarantees that the spectral information has converged at the same time. One of our preconditioning strategies makes use of these approximate eigenvectors (see Section 4.2.1) and then the performance of our preconditioner might depend on the quality of the eigenvectors. Consequently, we need to estimate the quality of the eigenvectors that will be checked via their backward error. The backward error associated with a normalized approximate eigenvector u of a matrix A [84] is defined by:

$$\min_{\Delta A}\left\{\xi > 0 : \|\Delta A\| \leq \xi\|A\| : (A+\Delta A)u = (u^H Au)u\right\} = \frac{\|Au - (u^H Au)u\|}{\|A\|}. \tag{2.12}$$

We suppose that the harmonic eigenvector g is normalized. In our context we have then to get a cheap upper-bound for

$$\frac{||AQ_m g - \rho Q_m g||_2}{||A||_2},$$

where $Q_m g$ is a normalized approximate eigenvector, and ρ is the Rayleigh quotient associated with $Q_m g$ that is defined by

$$\rho = (Q_m g)^H AQ_m g = g^H H_m g.$$

The residual norm $||AQ_m g - \rho Q_m g||_2$ can be computed using a formula from [62]:

$$||AQ_m g - \rho Q_m g||_2 = \sqrt{||(H_m - \rho I)g||_2^2 + |h_{m+1,m}|^2.|e_m^T g|^2}.$$

Algorithm 2 Right preconditioned restarted GMRES–DR algorithm: GMRES–DR(m,k).

1: Start
Choose m, the maximum size of the subspace, and k, the desired number of approximate eigenvectors. Choose an initial guess t_0 and compute $r_0 = b - AMt_0$. Let $v_1 = r_0/\|r_0\|$ and $\beta = \|r_0\|$.
2: First restart
Apply standard GMRES(m): generate Q_{m+1} and $\bar{H}_m$ with the Arnoldi iteration, solve $\min \|c - \bar{H}_m d\|$ for d, where $c = \beta e_1$, and form the new approximate solution $t_m = t_0 + Q_m d$. Let $\beta = h_{m+1,m}$, $t_0 = t_m$, and $r_0 = b - AMt_m$. Then compute k smallest harmonic eigenpairs $(\tilde{\theta}_i, \tilde{g}_i)$.
3: Orthonormalization of the first k vectors
Orthonormalize $\tilde{g}_i$'s, in order to form an $m \times k$ matrix P_k.
4: Orthonormalization of the $(k+1)^{-th}$ vector
First extend $p_1, \ldots, p_k$ (the columns of P_k) to length $(m+1)$ by appending a zero entry to each. Then orthonormalize the vector $c - \bar{H}_m d$ against them to form p_{k+1}. P_{k+1} is $(m+1) \times (k+1)$.
5: Form portions of new H and Q using the old H and Q
Let $\bar{H}_k^{\text{new}} = P_{k+1}^H \bar{H}_m P_k$ and $Q_{k+1}^{\text{new}} = Q_{m+1} P_{k+1}$. Then let $\bar{H}_k = \bar{H}_k^{\text{new}}$ and $Q_{k+1} = Q_{k+1}^{\text{new}}$.
6: Reorthogonalization of the $(k+1)^{-th}$ vector
Orthogonalize q_{k+1} against the preceding columns of Q_{k+1}^{new}.
7: Arnoldi iteration
Apply the Arnoldi iteration from q_{k+1} to form the rest of Q_{m+1} and $\bar{H}_m$. Let $\beta = h_{m+1,m}$.
8: Form the approximate solution
Let $c = Q_{m+1}^H r_0$ and solve $\min \|c - \bar{H}_m d\|$ for d. Let $t_m = t_0 + Q_m d$. Compute the residual vector $r = b - AMt_m = Q_{m+1}(c - \bar{H}_m d)$. Check $\|r\| = \|c - \bar{H}_m d\|$ for convergence. If it is the case $x_m = Mt_m$, if not proceed.
9: Eigenvalue computations
Compute the k smallest harmonic eigenpairs $(\tilde{\theta}_i, \tilde{g}_i)$
10: Restart
Let $t_0 = t_m$ and $r_0 = r$. Go to 3

We derive below an alternative expression to evaluate this residual norm.
From the definition (2.11) of g, we have:

$$\begin{aligned}
(H_m + h_{m+1,m}^2 f e_m^T) g &= g\theta \\
g^H (H_m + h_{m+1,m}^2 f e_m^T) g &= g^H g \theta \\
g^H H_m g &= \theta - h_{m+1,m}^2 (g^H f)(e_m^T g) \\
\rho &= \theta - h_{m+1,m}^2 (g^H f)(e_m^T g). \qquad (2.13)
\end{aligned}$$

Using equalities (2.11) and (2.13), the harmonic residual vector can be written as:

$$\begin{aligned}
AQ_m g - \rho Q_m g &= Q_m H_m g + h_{m+1,m} v_{m+1} (e_m^T g) - \rho Q_m g \\
&= \theta Q_m g - h_{m+1,m}^2 Q_m f (e_m^T g) + h_{m+1,m} v_{m+1} (e_m^T g) \\
&\quad - \theta Q_m g + h_{m+1,m}^2 (g^H f)(e_m^T g) Q_m g \\
&= Q_{m+1} \begin{bmatrix} h_{m+1,m}^2 (e_m^T g)((g^H f) g - f) \\ h_{m+1,m} (e_m^T g) \end{bmatrix}.
\end{aligned}$$

We finally obtain:

$$||AQ_m g - \rho Q_m g||_2 = |h_{m+1,m}| . |e_m^T g| \sqrt{|h_{m+1,m}|^2 . ||(g^H f) g - f||_2^{\,2} + 1}.$$

The quantity $||A||_2$ can be bounded below. From the Arnoldi relation (2.5) on the matrix A, it is easy to deduce that: $||H_m|| \leq ||A||$. Then the backward error associated with a normalized approximate eigenvector $Q_m g$ can be bounded above by:

$$\frac{||AQ_m - \rho Q_m g||_2}{||A||_2} \leq \frac{|h_{m+1,m}|.|e_m^T g|}{||H_m||_2} \sqrt{|h_{m+1,m}|^2.||(g^H f)g - f||_2{}^2 + 1}. \tag{2.14}$$

This calculation is of order $\mathcal{O}(m^2)$ and requires an estimation of the spectral norm of the Hessenberg matrix H_m. This computation is negligible since m remains small in comparison with the size of the matrix A.

2.3 The Flexible GMRES method

In 1993, Saad [74] introduced a variant of the GMRES method with variable right preconditioning. This algorithm enables the use of a different preconditioner at each step of the Arnoldi process. With a right preconditioner, the GMRES algorithm computes its iterates as

$$x_k = x_0 + MQ_k y_k, \tag{2.15}$$

where $y_k = \arg\min_y \|\beta e_1 - \bar{H}_k y\|$ and is based on the Arnoldi relation

$$AMQ_k = A[Mq_1, \cdots, Mq_k] = Q_{k+1}\bar{H}_k.$$

With variable preconditioners, a similar relation holds that can be written as:

$$A[M_1 q_1, \cdots, M_k q_k] = A[z_1, \cdots, z_k] = AZ_k = \mathcal{Q}_{k+1}\bar{\mathcal{H}}_k,$$

where $\mathcal{Q}_{k+1}$ has orthonormal columns and $\bar{\mathcal{H}}_k$ is upper Hessenberg. The algorithm generates iterates $x_k = x_0 + Z_k y_k$ that have minimal residual norm on the space $x_0 + span\{z_1, \cdots, z_k\}$. That is, with $y_k = \arg\min_y \|\beta e_1 - \bar{\mathcal{H}}_k y\|$. Because this GMRES variant allows for flexible preconditioners it is referred to as Flexible GMRES and we denote it by FGMRES. From an implementation point of view the main difference between right preconditioned GMRES and FGMRES is the memory requirement. In that latter algorithm, both $\mathcal{Q}_k$ and Z_k need to be stored. Furthermore, while only happy breakdowns might occur in GMRES, FGMRES can break down at some step before it has computed the solution. We describe this method in Algorithm 3 and refer to [74] for a complete description of the convergence theory.

As for the GMRES algorithm, we use an implementation of the FGMRES algorithm with a reverse communication mechanism [32], also available via the same Web address:

http://www.cerfacs.fr/algor/Softs/

The AS_ELFIP code uses an out-of-core version of this package.

2.4 The Seed-GMRES method

The Seed-GMRES [80] can be used for solving a sequence of linear systems with the same coefficient matrix. The idea consists in benefiting from the current solve at each restart by using the associated Krylov space to compute a better initial guess for all of the other linear systems. For the sake of simplicity of the notation, we describe below the method for the solution of only two right-hand sides. The problem for a right preconditioned version can be written:

$$\begin{cases} AM(t^1, t^2) = (b^1, b^2) \text{ with,} \\ (t_0^1, t_0^2) \text{ initial guesses.} \end{cases}$$

Algorithm 3 Right preconditioned restarted FGMRES algorithm: FGMRES(m).

Choose a convergence threshold ε ;
Choose an initial guess x_0 ;
$r_0 = b - Ax_0$;
for $s = 1, \ldots$ **do**
  $\beta = \|r_0\|$;
  $q_1 = r_0/\beta$;
  for $j = 1, \ldots, m$ **do**
    $z_j = M_j q_j$
    $w = Az_j$;
    for $i = 1$ `to` j **do**
      $h_{i,j} = q_i^H w$;
      $w = w - h_{i,j} q_i$;
    end for
    $h_{j+1,j} = \|w\|$;
    $q_{j+1} = w/h_{j+1,j}$;
    Find y_j the least-squares solution of $\min_y \|\beta e_1 - \bar{H}_j y\|$
    if (stopping criterion $\leq \varepsilon$) **then**
      $x = x_0 + Z_j y_j$;
      stop ;
    end if
  end for
  $x_0 = x_0 + Z_m y_m$
  $r_0 = b - Ax_0$;
end for

The seed GMRES method starts by solving the first system $AMt^1 = b^1$ using the GMRES algorithm. It computes an orthonormal basis of the Krylov space Q^1_{m+1} and the upper-Hessenberg matrix $\bar{H}^1_m$ such that $AMQ^1_m = Q^1_{m+1}\bar{H}^1_m$. At each restart of the current solve, the initial guess t^2_0 is updated by minimizing the norm of the residual $r^2_0 = b^2 - AMt^2_0$ of the second system on the current basis Q^1_{m+1}. We search for a new initial guess $\hat{t}^2_0$ so that:

$$\left\{ \begin{array}{l} \hat{t}^2_0 = t^2_0 + Q^1_m u^2 \text{ with,} \\ u^2 = \underset{u \in \mathbb{C}^m}{\arg\min} ||b^2 - AM(t^2_0 + Q^1_m u)||_2. \end{array} \right.$$

Using the fact that:

$$\begin{aligned} ||b^2 - AM(t^2_0 + Q^1_m u)||^2_2 &= ||r^2_0 - AMQ^1_m u||^2_2 \\ &= ||(I_n - Q^1_{m+1}(Q^1_{m+1})^T) r^2_0 + Q^1_{m+1}(Q^1_{m+1})^T r^2_0 - AMQ^1_m u||^2_2 \\ &= ||(I_n - Q^1_{m+1}(Q^1_{m+1})^T) r^2_0 + Q^1_{m+1}((Q^1_{m+1})^T r^2_0 - \bar{H}^1_m u)||^2_2 \\ &= ||(I_n - Q^1_{m+1}(Q^1_{m+1})^T) r^2_0||^2_2 + ||Q^1_{m+1}((Q^1_{m+1})^T r^2_0 - \bar{H}^1_m u)||^2_2 \\ &= ||(I_n - Q^1_{m+1}(Q^1_{m+1})^T) r^2_0||^2_2 + ||(Q^1_{m+1})^T r^2_0 - \bar{H}^1_m u||^2_2, \end{aligned}$$

we finally should solve at each restart the least-squares problem: $\arg\min_u ||(Q^1_{m+1})^T r^2_0 - \bar{H}^1_m u||_2$, where we already have a QR factorization of $\bar{H}^1_m$ built by the GMRES algorithm. Once the first system has been solved, we simply run a new GMRES method for the second right-hand side using the new initial guess. For a sequence of nb right-hand sides, during the solve of the current system ℓ , we have to update at each restart the subsequent $(nb - \ell + 1)$ initial guesses. In order to take account of all the updates of all guesses, extra storage of nb vectors is required for the nb initial guesses. The Seed-GMRES algorithm is described in Algorithm 4 for nb right-hand sides.

Algorithm 4 Right preconditioned restarted Seed-GMRES algorithm: Seed-GMRES(m).

1: Choose a convergence threshold ε ;
2: **for** $\ell = 1, \ldots, nb$ **do**
3: Choose an initial guess t_0^ℓ ;
4: $r_0^\ell = b^\ell - AMt_0^\ell$;
5: **for** $s = 1, \ldots$ **do**
6: $\beta = \|r_0^\ell\|$;
7: $q_1^\ell = r_0^\ell / \beta$;
8: **for** $j = 1, \ldots, m$ **do**
9: $w = AMq_j^\ell$;
10: **for** $i = 1$ `to` j **do**
11: $h_{i,j} = (q_i^\ell)^H w$;
12: $w = w - h_{i,j} q_i^\ell$;
13: **end for**
14: $h_{j+1,j} = \|w\|$;
15: $q_{j+1}^\ell = w / h_{j+1,j}$;
16: Find y_j of the least-squares solution of $\min_y \|\beta e_1 - \bar{H}_j y\|$
17: **if** (stopping criterion $\leq \varepsilon$) **then**
18: $t = t_0 + Q_j^\ell y_j$;
19: goto 29 ;
20: **end if**
21: **end for**
22: $t_0^\ell = t_0^\ell + Q_m^\ell y_m$;
23: $r_0^\ell = b - AMt_0^\ell$;
24: **for** $q = \ell + 1, \ldots, nb$ **do**
25: $u^q = \arg\min_u ||b^q - AM(t_0^q + Q_m^\ell u)||_2$;
26: $t_0^q = t_0^q + Q_m^\ell u^q$;
27: **end for**
28: **end for**
29: $x^\ell = Mt$;
30: **end for**

2.5 Some computational aspects in the implementation of the solvers

2.5.1 Stopping criterion

We have chosen to base the stopping criterion for any GMRES type linear solver on the normwise backward error [22, 47, 70]. Backward error analysis, introduced by Wilkinson [89], is an appealing concept for analysing the quality of an approximate solution:

1. it is independent from the details of round-off propagation: the errors introduced during the computation are interpreted in terms of perturbations of the initial data, and the computed solution is considered as exact for the perturbed problem;

2. because round-off errors are seen as data perturbations, they can be compared with errors due to numerical approximations (consistency of numerical schemes) or to physical measurements (uncertainties on data coming from experiments for instance) or to the FMM accuracy in our framework.

The backward error measures in fact the distance between the data of the initial problem and that of the perturbed problem. In the context of linear systems, classical choices are normwise and

componentwise perturbations [22, 47]. These choices lead to explicit formulae for the backward error (often a normalized residual) which is then easily evaluated. For iterative methods, it is generally admitted that the normwise model of perturbation is appropriate [8, 33].

Let $\tilde{x}$ be an approximate solution of $x = A^{-1}b$. The relative normwise backward error associated with $\tilde{x}$, considering a perturbation ΔA on A and a perturbation Δb on b, can be defined as:

$$\begin{aligned}\eta_{A,b}(\tilde{x}) &= \min_{\Delta A,\Delta b}\{\xi > 0;\ \|\Delta A\| \le \xi\|A\|,\ \|\Delta b\| \le \xi\|b\| \text{ and } (A+\Delta A)\tilde{x} = b+\Delta b\} \\ &= \frac{\|A\tilde{x}-b\|}{\|A\|\|\tilde{x}\|+\|b\|}.\end{aligned}$$

The relative normwise backward error associated with $\tilde{x}$, considering only a perturbation Δb on b, can be defined as:

$$\begin{aligned}\eta_b(\tilde{x}) &= \min_{\Delta b}\{\xi > 0;\ \|\Delta b\| \le \xi\|b\| \text{ and } A\tilde{x} = b+\Delta b\} \\ &= \frac{\|A\tilde{x}-b\|}{\|b\|}.\end{aligned} \tag{2.16}$$

The best we can require from an algorithm is a backward error of the order of the machine precision. In practice, the approximate solution is acceptable when its backward error is of the order of the uncertainty in the data. Therefore, there is often no gain in iterating after the backward error has reached machine precision or data accuracy. Either $\eta_{A,b}$ or η_b are recommended for implementing a stopping criterion strategy. Nevertheless, implementing the stopping criterion $\eta_b(\tilde{x}) \le \varepsilon$ is far simpler than for $\eta_{A,b}(\tilde{x}) \le \varepsilon$, as we do not need to estimate the norm of A. That is why we choose η_b for the stopping criterion for all our experiments with GMRES and its variants.

The packages [32, 33] implement a stopping criterion based on the backward error associated with the preconditioned linear system and not with the original linear system, to comply with the backward stability argument for the GMRES method given in [42]. Furthermore, for a left preconditioner, the residual which is readily available in the algorithm is that of the preconditioned system. It would be too expensive to compute the residual of the unpreconditioned system at each iteration. It should be pointed out that the stopping criterion $\eta_b(\tilde{x})$ associated with the preconditioned system $MAx = Mb$ depends on the chosen preconditioner. This is no longer true for a right preconditioner where the stopping criterion $\eta_b(\tilde{t})$ associated with the preconditioned system $AMt = b$ with $x = Mt$, can be written:

$$\eta_b(\tilde{t}) = \frac{\|AM\tilde{t}-b\|}{\|b\|} = \frac{\|A\tilde{x}-b\|}{\|b\|} = \eta_b(\tilde{x}).$$

Consequently, it does not depend on the chosen preconditioner. This is the reason why right preconditioning is often preferred in many applications. Moreover, as we plan to compare the efficiency of different preconditioners on the same problem in the following chapters, we have to select a stopping criterion independent of the chosen preconditioner. For this reason we choose right preconditioners for all our numerical experiments.

In electromagnetism simulations, the stopping criterion consists in reducing $\eta_b(\tilde{x})$ down to 10^{-3}. Although this value may appear to be large, it is sufficient to obtain accurate radar cross section results.

2.5.2 Evaluation of the norm of the residual

At each step j of any GMRES-like method, one needs to solve the least-squares problem (2.6). The matrix $\bar{H}_j$ involved in this least-squares problem is a $(j+1)\times j$ complex matrix which is

upper Hessenberg. We make the same choice as in [33]. The solution of (2.6) will be based on the QR factorization of the matrix $[\bar{H}_j,\ \beta e_1]$: if $QR = [\bar{H}_j,\ \beta e_1]$ where Q is a orthonormal matrix and $R = (r_{ik})$ is an $(j+1) \times (j+1)$ upper triangular matrix. The solution y_j of (2.6) is given by:

$$y_j = R(1:j, 1:j)^{-1} R(1:j, j+1). \tag{2.17}$$

Here, $R(1:j, 1:j)$ denotes the first $j \times j$ submatrix of R and $R(1:j, j+1)$ stands for the last column of R. The 2–norm of the residual is a by-product of the solution of the least-squares problem:

$$\|r_j\|_2 = \|b - Ax_j\|_2 = \|\beta e_1 - \bar{H}_j y_j\|_2 = |r_{j+1,j+1}|. \tag{2.18}$$

However, it is well known that, in finite-precision arithmetic, the norm of the computed residual $|r_{j+1,j+1}|$ given by the Arnoldi process may differ significantly from the norm of the true residual $||r_j||_2$. From our numerical experiments, we follow the strategy implemented in [32, 33]. Let $\tilde{\eta}_b(\tilde{x})$ be the normwise backward error associated with $\tilde{x}$ and defined as:

$$\tilde{\eta}_b(\tilde{x}) = \frac{|r_{j+1,j+1}|}{\|b\|}. \tag{2.19}$$

As soon as $\tilde{\eta}_b(\tilde{x})$ becomes less than the prescribed tolerance ε, we iterate until $\eta_b(\tilde{x})$ becomes itself lower than the tolerance.

2.5.3 Computation of the orthonormal basis Q_m

The quality of the orthogonality of the Q_j plays a central role in GMRES as deteriorating it might delay or slow down the convergence. On the other hand, ensuring very good orthogonality might be expensive and useless for some applications. Consequently a trade-off has to be found to balance the numerical efficiency of the orthogonalization scheme and its inherent implementation efficiency on a given target computer.

It is well known that the classical Gram-Schmidt algorithm (CGS) is numerically worse than the modified Gram-Schmidt algorithm (MGS) [13]. But the CGS algorithm can be implemented in an efficient manner by gathering the dot products into one matrix–vector product. In finite-precision arithmetic, there might be a severe loss of orthogonality in the computed basis; this loss can be compensated by selectively iterating the orthogonalization scheme. The resulting algorithm is called iterative classical Gram-Schmidt (ICGS). It is particularly attractive in a parallel distributed environment, where the global reduction involved in the computation of the dot product is a well-known bottleneck [31, 34, 56, 79]. An iterative modified Gram-Schmidt (IMGS) results in an algorithm of the same numerical quality as ICGS [38], but its main drawback is the increased number of dot products. For our numerical experiments, we select the ICGS algorithm [32, 33].

Chapter 3

One level preconditioning techniques for integral equations

The design of robust preconditioners for boundary integral equations can be challenging. Simple preconditioners like the diagonal of A, diagonal blocks, or a band can be effective only when the coefficient matrix has some degree of diagonal dominance depending on the integral formulation [82]. Block diagonal preconditioners are generally more robust than their point-wise counterparts, but may require matrix permutations or renumbering of the grid points to cluster the large entries close to the diagonal. Incomplete factorizations have been successfully used on nonsymmetric dense systems in [78] and hybrid integral formulations in [55], but on the EFIE the triangular factors computed by the factorization are often very ill-conditioned due to the indefiniteness of A [19].

Approximate inverse methods are generally less prone to instabilities on indefinite systems, and several preconditioners of this type have been proposed in electromagnetism (see for instance [1, 23, 54, 77, 87]). Owing to the rapid decay of the discrete Green's function, the location of the large entries in the inverse matrix exhibits some structure. In addition, the entries of large magnitude represent only a very small number compared to the others. This means that a very sparse matrix is likely to retain the most relevant contributions to the exact inverse. This desirable property can be effectively exploited in the design of robust approximate inverses in electromagnetism.

3.1 Frobenius-norm minimization preconditioner

3.1.1 General description

In this section we describe an approximate inverse preconditioner based on Frobenius-norm minimization. The original idea, due to Benson and Frederickson [10, 35], 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). The independence of these least-squares problems follows immediately from the identity:

$$\|I - MA\|_F^2 = \|I - AM^T\|_F^2 = \sum_{j=1}^{n} \|e_j - Am_{j\bullet}\|_2^2 \tag{3.1}$$

where e_j is the j^{-th} canonical unit vector and $m_{j\bullet}$ is the column vector representing the j^{-th} row of M. In the case of right preconditioning, the analogous relation

$$\|I - AM\|_F^2 = \sum_{j=1}^{n} \|e_j - Am_{\bullet j}\|_2^2 \tag{3.2}$$

holds, where $m_{\bullet j}$ is the column vector representing the j^{-th} 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, that is the set of indices

$$\mathcal{S} = \{\ (i,j)\ \subseteq\ [1,n]^2 \mid m_{ij} = 0\ \}. \tag{3.3}$$

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. Two different approaches can be followed for this purpose, that is 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.

Adaptive strategies can solve fairly general or hard problems but tend to be very expensive. The use of effective static pattern selection strategies can greatly reduce the amount of work in terms of CPU-time, and improve substantially the overall setup process, introducing significant scope for parallelism. On boundary integral equations the discrete Green's function (1.1.2) decays rapidly far from the diagonal, and the inverse of A may have a very similar structure to that of A. The discrete Green's function can be considered as a row or as a column of the exact inverse depicted on the physical computational grid. In this case, a good pattern for the preconditioner can be computed in advance using graph information from $\tilde{A}$, a sparse approximation of the coefficient matrix constructed by dropping all the entries lower than a prescribed global threshold [1, 17, 49]. When fast methods are used for the matrix-vector products, all the entries of A are not available in memory and the pattern can be formed by using the near-field part of the matrix that is explicitly computed and available in the FMM [54].

Since we work in an integral equation context, relevant information for the construction of the pattern of M can be extracted from the triangular mesh. Each degree of freedom (DOF), representing an unknown in the linear system, corresponds to an edge. The sparsity pattern for any column of the preconditioner can be defined according to the concept of level k neighbours. Level 1 neighbours 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 neighbours are all the level 1 neighbours plus the DOFs in the triangles that are neighbours of the two triangles considered at level 1, and so forth. When the object geometries are smooth, only the neighbouring edges in the mesh can have a strong interaction with each other, while far-away connections are generally much weaker. Thus an effective pattern for the j^{-th} column of the approximate inverse can be computed by selecting in the mesh the edge j and its q^{-th} level nearest-neighbours. 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 [17]. When the object geometries are not smooth or have disconnected parts, far-away edges in the mesh can have a strong interaction with each other and be strongly coupled in the inverse matrix. In this case a more robust pattern for the preconditioner can be computed using physical information, that is selecting for each edge all those edges within a sufficiently large geometric neighbourhood. A comparison of the pattern selection strategies based both on algebraic and mesh information on a large set of problems is described in [17]. It has been found that those exploiting geometric information are the most effective at capturing the large entries of the inverse. We refer to the Frobenius-norm minimization method described in this section and computed using a static pattern strategy based on geometric information as the M_{Frob} preconditioner.

The construction of M_{Frob} costs $\mathcal{O}(n^2)$ arithmetic operations. This cost can be significantly reduced if the preconditioner is computed using as input a sparse approximation $\tilde{A}$ of the dense

coefficient matrix A. On general problems, this approach can cause a severe deterioration of the quality of the preconditioner. In an integral equation context it is likely to be more effective because the boundary element method generally introduces a very localized strong coupling among the edges in the underlying mesh. It means that a very sparse matrix can still retain the most relevant contributions from the singular integrals that give rise to dense matrices. If the sparsity pattern of M is known in advance, the nonzero structure for the j^{-th} column of M is automatically determined, and defined as

$$J = \{i \in [1,n] \; s.t. \; (i,j) \in \mathcal{S}\}.$$

The least-squares solution involves only the columns of $\tilde{A}$ indexed by J; we indicate this subset by $\tilde{A}(:,J)$. When $\tilde{A}$ is sparse, many rows in $\tilde{A}(:,J)$ are usually null, not affecting the solution of the least-squares problems (3.1). Thus if I is the set of indices corresponding to the nonzero rows in $\tilde{A}(:,J)$, and if we define by $\hat{A} = \tilde{A}(I,J)$, by $\hat{m}_j = m_j(J)$, and by $\hat{e}_j = e_j(J)$, the actual "reduced" least-squares problems to solve are

$$min\|\hat{e}_j - \hat{A}\hat{m}_j\|_2, j = 1, ..., n. \tag{3.4}$$

Usually problems (3.4) have much smaller size than problems (3.1) and can be effectively solved by dense QR factorization. In [1] 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. Up to a certain limit, selecting more entries in M than in $\tilde{A}$ can enable the computation of a more robust preconditioner, and the additional cost in terms of CPU time is negligible because of the complexity of the QR factorization used to solve the least-squares problems [17]. Increasing the number of rows in problem (3.4), that is the number of entries of $\tilde{A}$, is much cheaper in terms of overall CPU time than increasing the density of the preconditioner, that is the number of columns in the least-squares problems.

3.1.2 Implementation of the preconditioner in the fast multipole context

The box-wise decomposition of the domain naturally leads to an *a priori* pattern selection strategy for M and $\tilde{A}$ in the FMM context. We adopt the following criterion: the nonzero structure of the column of the preconditioner associated with a given edge is defined by retaining all the edges within its leaf box and those within one level of neighbouring boxes. The structure for the sparse approximation of the dense coefficient matrix is defined by retaining the entries associated with edges included in the given leaf box as well as those belonging to the two levels of neighbours. The approximate inverse has a sparse block structure; each block is dense and is associated with one leaf box. Indeed the least-squares matrices corresponding to edges within the same leaf box are identical because they are defined using the same nonzero structure and the same set of entries of A. It means that we only have to compute one QR factorization per leaf box. Consequently, the numerical complexity to build the preconditioner is linear with the number of leaf boxes. In our implementation we use two different octrees, and thus two different partitionings, to assemble the preconditioner and to compute the matrix-vector product via the FMM. The size of the smallest boxes in the partitioning associated with the preconditioner is a user-defined parameter that can be tuned to control the number of nonzeros computed per column. According to our criterion, the larger the size of the leaf boxes, the larger the geometric neighbourhood that determines the sparsity structure of the columns of the preconditioner. The parallel implementation for building the preconditioner consists in assigning disjoint subsets of leaf boxes to different processors and performing the least-squares solutions independently on each processor. At each step of the Krylov solver, applying the preconditioner is easily parallelizable as it reduces to a regular matrix-vector product [85].

3.1.3 Numerical experiments

As already mentioned, all the numerical experiments arising from electromagnetism calculations in this thesis use a threshold equal to 10^{-3} for the stopping criterion based on $\eta_b(\tilde{x})$. This tolerance is accurate for engineering purposes, as it enables the correct construction of the radar cross section of the objects. The initial guess for all GMRES type solvers except the seed-GMRES is the zero vector.

In this section, we present the numerical behaviour of the Frobenius-norm minimization preconditioner on many sizes of the target problems. Those results have been extracted from [20] but for the sake of comparison with other techniques we think that it is worth summarizing the main ones below. The size of the problems is varied by increasing the frequency of the illuminating wave, while the number of points per wavelength remains constant, equal to ten [9]. The leaf box dimension in the partitioning associated with the preconditioner keeps the same value for increasing frequency. This means that the number of nonzeros per column in the preconditioner is constant and independent of the problem size. Consequently, the density of M_{Frob} decreases for increasing problem size. We choose for each geometry described in Figure 1.7, a right-hand side, associated with an incident wave, giving rise to a linear system among the most difficult to solve. The illuminating direction (θ, ϕ) for the Cetaf case is $(65^o, 0^o)$, for the Aircraft case $(30^o, 30^o)$, for the Cobra case $(25^o, 0^o)$ and for the Almond case $(90^o, 40^o)$. The choice of the angle per geometry is fixed for all numerical experiments in this chapter. All the runs have been performed on eight processors of a Compaq Alpha server.

In Table 3.1 we display for each geometry the size of the problems, the density and the construction time of M_{Frob}, the number of matrix-vector products and the elapsed time required to converge using full-GMRES or GMRES(120) preconditioned with M_{Frob}. We see that the numerical behaviour of M_{Frob} does not scale well with the size of the problem, except for the Cobra case. Using full-GMRES, the Aircraft case of size 591 900 and the Almond case of size 943 137 exceed the memory limit and the time limit allocated to the runs performed with 8 processors. The orthogonalization involved in the Arnoldi procedure strongly affects the solution time. Good illustrations of this phenomenon are the two largest Cetaf cases: the elapsed time to obtain the solution with full-GMRES is twice as great as with GMRES(120), while the number of matrix-vector products with full-GMRES is half that for GMRES(120). Provided we get convergence, the use of a restarted GMRES often reduces the solution time even though it significantly deteriorates the convergence. There is no convergence on the largest aircraft problems after 2000 iterations of GMRES(120). The preconditioner becomes less and less effective as the size of the problem increases because it becomes sparser. The number of unknowns per box remains constant, but the number of boxes increases leading to a decrease in the density of the preconditioner and a reduced efficiency. We investigate now the influence of the density of M_{Frob} on the numerical results.

The size of the leaf boxes in the oct-tree associated with the preconditioner is increased in the Cobra case of size 179 460, using GMRES(120) and the same right-hand side (illuminating direction $(\theta, \phi) = (25^o, 0^o)$). In Table 3.2, we display the number of matrix-vector products required to achieve convergence, the elapsed time for the construction of M_{Frob}, the solution time and the overall time to deal with one right-hand side, when the density of M_{Frob} varies. The size of the leaf box is denoted by "Radius"; we have to multiply it by the wavelength to obtain its physical size. Increasing the number of nonzeros per column tends usually to decrease the number of iterations. Going from the density 0.039 to the density of 0.358, that is nine times larger, only reduces by a factor of 1.5 the number of matrix-vector products. On top of that, the corresponding construction time is multiplied by 22 and becomes larger than the solution time. The size of the least-squares problem increases very much. Then a trade-off between the construction cost and the numerical performance has to be found. In most of the cases it is obtained for a size of leaf box equal to 0.12 wavelengths. This is the default value set in the AS_ELFIP code. If the preconditioner is used to solve systems with the same coefficient matrix and multiple right-hand sides, it might be worth computing more nonzeros if we have enough disk space, because the construction cost can

be quickly amortized. However, significantly enlarging the density is not feasible on the largest problems because we would exceed the memory and disk capacity of our computer. For more details on the numerical behaviour of M_{Frob}, we refer to [16, 20]. In order to solve very large problems, we investigate other preconditioning approaches in the rest of this manuscript.

Aircraft						
Size	Density	Setup time	GMRES(∞)		GMRES(120)	
			# Mat.V	Time	# Mat.V	Time
94 704	0.28	11mn	746	2h 09mn	1956	3h 13mn
213 084	0.13	31mn	973	7h 19mn	*+2000*	*> 7h 56mn*
591 900	0.09	1h 30mn	1461	16h 42mn$^{(64)}$	*+2000*	*> 1d 57mn*
Almond						
Size	Density	Setup time	GMRES(∞)		GMRES(120)	
			# Mat.V	Time	# Mat.V	Time
104 793	0.19	6mn	234	20mn	253	17mn
419 172	0.05	21mn	413	2h 44mn	571	2h 26mn
943 137	0.02	49mn	454	3h 35mn$^{(32)}$	589	5h 55mn
Cetaf						
Size	Density	Setup time	GMRES(∞)		GMRES(120)	
			# Mat.V	Time	# Mat.V	Time
48 519	0.32	3mn	499	33mn	521	21mn
134 775	0.11	6mn	618	1h 45mn	1125	1h 55mn
264 156	0.06	13mn	710	9h	1373	4h 46mn
531 900	0.03	20mn	844	1d 18mn	1717	14h 08mn
Cobra						
Size	Density	Setup time	GMRES(∞)		GMRES(120)	
			# Mat.V	Time	# Mat.V	Time
60 695	0.24	2mn	369	26mn	516	23mn
179 460	0.09	7mn	353	1h 11mn	406	1h 02mn

Table 3.1: Number of matrix-vector products and elapsed time to achieve converge for the four geometries using full-GMRES and GMRES(120), preconditioned with M_{Frob}. All these results are obtained on 8 processors, except those marked with (k), that were run on k processors.

3.2 Exact inverse versus approximate inverse

The governing ideas that led to the study and implementation of the preconditioner M_{Frob} were its relatively good numerical efficiency and its embarrassingly parallel implementation both in its construction and in its application to a vector. Its construction from a sparse approximation $\tilde{A}$ of A reduces to the solution of a set of small independent linear least-squares and its application is just a sparse matrix-vector product. A natural concern that arises is how much have we lost in terms of the numerical efficiency of the preconditioner by using an approximate inverse rather than an "exact inverse" of $\tilde{A}$. To address this question and because we are still interested in large parallel distributed computation we select the MUMPS package [3, 4] to perform the LDL^T factorization of $\tilde{A}$ and refer to the associated preconditioner as M_{Mumps}.

Radius	Density (%)	# Mat.V in GMRES(120)	Construction Time	Solution Time	Overall Time
0.08	0.039	430	4mn	1h	1h 04mn
0.10	0.062	383	5mn	50mn	55mn
0.12	0.091	406	7mn	49mn	56mn
0.14	0.120	380	11mn	48mn	59mn
0.18	0.204	351	31mn	52mn	1h 23mn
0.24	0.358	280	1h 22mn	43mn	2h 05mn

Table 3.2: Number of matrix-vector products and elapsed time to build M_{Frob} and to obtain the solution using preconditioned GMRES(120) on a Cobra problem of size 179 460, varying the radius monitoring the density of the preconditioner. The tests were run on 8 processors of the Compaq machine.

3.2.1 Brief overview of the parallel distributed sparse direct solver: MUMPS

The MUltifrontal Massively Parallel Solver (MUMPS, [3, 4]) is a parallel distributed sparse direct solver. The matrix to be factorized can be either symmetric positive definite, general symmetric or general unsymmetric, in complex or real arithmetic. The MUMPS software uses a multifrontal technique to perform either an LU or an LDL^T factorization of the matrix. It implements three phases: symbolic analysis, numerical factorization and solution. During the symbolic analysis phase some preprocessing is applied in order to permute the matrix in an attempt to decrease the number of nonzeros in the factors. The factorization phase tries to follow the decision of the analysis but numerical pivoting is applied to ensure as much as possible a stable factorization. The independence of the computation arising from the elimination tree and in the dense calculation in the frontal matrices are efficiently exploited to achieve good performance on a distributed memory parallel computer. The solution phase performs a forward and backward substitution on the computed factors. The software is written in Fortran 90 and a C interface is available. We use this latter interface to call MUMPS in the AS_ELFIP code. The parallel version of MUMPS requires MPI for message passing and makes use of BLAS, BLACS and ScaLAPACK libraries. We refer the reader to the Users' Guide [2] for more details. Many orderings are already available in the MUMPS package, and we choose to use the METIS package [48].

3.2.2 Choice of the pattern for the M_{Mumps} preconditioner

The exact factorization is computed for the sparse approximation $\tilde{A}$ of the dense coefficient matrix A. As previously defined in Section 3.1.2, the structure for $\tilde{A}$ is defined by retaining the entries associated with edges included in the given leaf box of the oct-tree as well as the edges belonging to the neighbours and the neighbours of the neighbours (level two neighbours). The preconditioner M_{Frob} is classically built from two levels of neighbours for $\tilde{A}$. For M_{Mumps} we investigate three possibilities: using an approximation $\tilde{A}$ obtained from zero level, one and two levels of neighbours. We notice that the zero-level variant reduces to a block diagonal preconditioner, where each diagonal block is associated with one leaf box.

In Figure 3.1 we depict the pattern of upper triangular part of $\tilde{A}$ (resp. the lower triangular part) obtained using zero level of neighbours (resp. using one level of neighbours) on the Cetaf 5 391 example. The zero level produces a block diagonal pattern, each new level of neighbours adds extra off-diagonal blocks. Because we consider the EFIE formulation, $\tilde{A}$ is symmetric then we perform a LDL^T factorization.

Because all the data structures of MUMPS are in-core while the AS_ELFIP code uses out-of-core functionalities intensively, a comparison between the performance of M_{Frob} and M_{Mumps} in terms

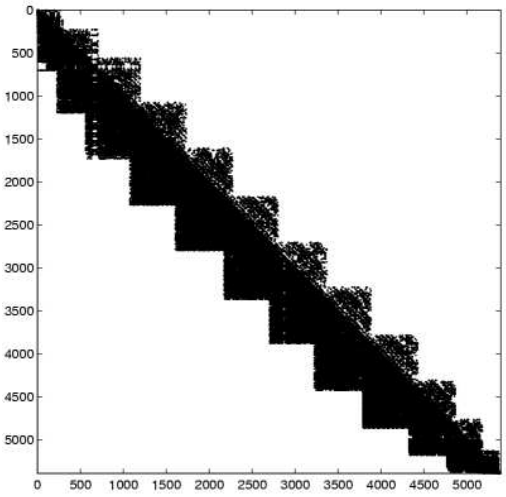


Figure 3.1: Cetaf 5 391 - upper triangular part corresponds to zero level of neighbours, lower triangular part corresponds to one level of neighbours.

of elapsed time would be unfair. For this reason the elapsed times are only given for information and certainly not for comparison. In the sequel, the two approaches are mainly compared in terms of floating-point operations to build and apply the preconditioners. This latter cost is directly related to the number of nonzero entries in the L and D factors computed by M_{Mumps} and in the M_{Frob} matrix.

3.2.3 Cost of the construction of M_{Mumps}

For one level of neighbours in $\tilde{A}$, we report in Table 3.3 the performance of MUMPS in terms of millions of nonzero entries in the factors, floating-point operations (GFlop) and elapsed time to perform the factorization. For the sake of comparison, we also display these quantities for M_{Frob}. On the Almond of size 419 172, we see that the number of nonzeros in the factors computed by MUMPS is five time larger than the number of nonzero in M_{Frob}. This leads to significant increase in the memory requirement. The cumulated memory space needed for the factors on all processors is about 25 Giga Bytes (GB) for this example. Similarly, the largest Cetaf requires 17 GB, the largest Aircraft needs 9 GB and the largest Cobra requires 6 GB. Both the largest Aircraft and Almond have been run on more than 8 processors so that enough memory was available for the factors (1 GB per processor on the Compaq Alpha Server). Concerning the number of floating-point operations in the setup phase of the preconditioner, it is not surprising that building M_{Mumps} is cheaper than M_{Frob}; computing the inverse of a matrix by solving a sequence of least-squares problems is not a very efficient approach. Nevertheless, because M_{Frob} is built using only dense linear algebra kernels (mainly level 3 BLAS), we note that the sustained MFlop rate is higher for M_{Frob} than for M_{Mumps}.

In Table 3.4, we display the memory required by MUMPS if the number of levels of neighbours is changed from one to two in the definition of $\tilde{A}$. The column entitled "GB" represents the amount of distributed memory space required to store the factor computed by MUMPS . We see that for problems of moderate size the amount of memory required to compute and store the factors becomes fairly large when two levels of neighbours are considered. In particular, for the Cetaf of size 86 256, we had to perform the experiments on 63 processors in order to have enough memory space to perform the factorization.

Aircraft							
	Millions of nonzero entries			Construction		Construction	
		MUMPS		GFlop		Time	
Size	$nnz(M_{Frob})$	$(nnz(\tilde{A})+n)/2$	$nnz(L)+nnz(D)$	M_{Frob}	M_{Mumps}	M_{Frob}	M_{Mumps}
94 704	24.9	12.5	49.1	5 054.6	27.4	11mn	26s
213 084	58.7	29.5	133.9	15 829.7	105.6	15mn $^{(16)}$	56s $^{(16)}$

Almond							
	Millions of nonzero entries			Construction		Construction	
		MUMPS		GFlop		Time	
Size	$nnz(M_{Frob})$	$(nnz(\tilde{A})+n)/2$	$nnz(L)+nnz(D)$	M_{Frob}	M_{Mumps}	M_{Frob}	M_{Mumps}
104 793	21.0	10.5	86.9	2 578.7	97.7	6mn	1mn
419 172	83.5	41.9	417.2	9 918.3	813.8	8mn $^{(20)}$	4mn $^{(20)}$

Cetaf							
	Millions of nonzero entries			Construction		Construction	
		MUMPS		GFlop		Time	
Size	$nnz(M_{Frob})$	$(nnz(\tilde{A})+n)/2$	$nnz(L)+nnz(D)$	M_{Frob}	M_{Mumps}	M_{Frob}	M_{Mumps}
86 256	13.3	6.6	47.7	1 558.8	38.5	5mn	27 s
134 775	19.7	9.9	79.1	19 737.0	73.9	6mn	48 s
264 156	39.4	19.8	172.4	4 593.3	210.5	14mn	2mn

Cobra							
	Millions of nonzero entries			Construction		Construction	
		MUMPS		GFlop		Time	
Size	$nnz(M_{Frob})$	$(nnz(\tilde{A})+n)/2$	$nnz(L)+nnz(D)$	M_{Frob}	M_{Mumps}	M_{Frob}	M_{Mumps}
60 695	8.8	4.4	30.1	690.8	19.8	2mn	16 s
179 460	29.2	14.7	122.5	2 787.3	141.2	8mn	1mn

Table 3.3: Construction phase. All these results are obtained on 8 processors except those marked with $^{(k)}$, that were run on k processors.

	One level		Two levels	
	$nnz(L)+nnz(D)$	GB of RAM	$nnz(L)+nnz(D)$	GB of RAM
Aircraft 23 676	8.8	0.8	19.1	2
Cobra 60 695	30.1	2	92.5	7
Cetaf 86 256 $^{(63)}$	47.7	5	181.1	36

Table 3.4: Effect of the level of neighbours to compute $\tilde{A}$ on size of the factors computed by MUMPS . All the experiments have been performed on eight processors except the Cetaf that required the use of 63 processors.

One way to reduce the size of the factors is to sparsify the approximation of A constructed using the neighbouring approach. In that context, we discard all the entries of $\tilde{A}$ that are small in magnitude and build a sparse approximation denoted by $\tilde{A}_\tau$. More precisely, to preserve the symmetry of $\tilde{A}_\tau$ we apply the following strategy:

$$\begin{cases} \tilde{A}_\tau(i,j) = \tilde{A}(i,j) & \text{if } |\tilde{A}(i,j)| > \tau * max(|\tilde{A}(i,i)|, |\tilde{A}(j,j)|, \\ \tilde{A}_\tau(i,j) = 0 & \text{otherwise.} \end{cases}$$

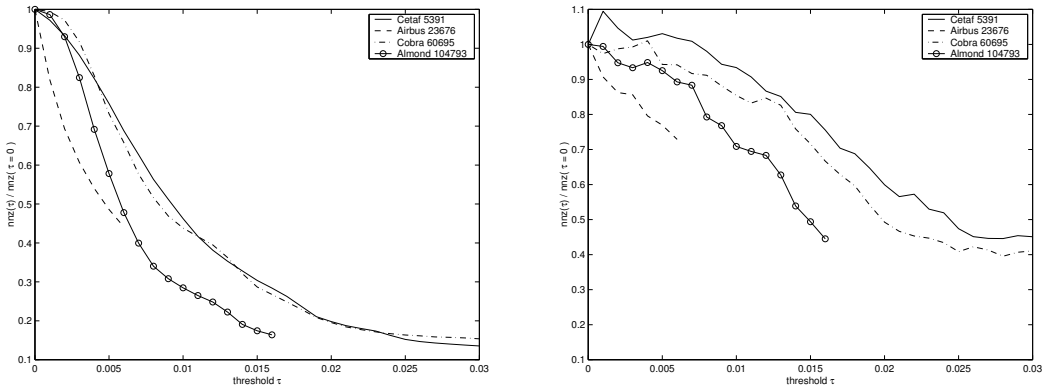
We display in Figure 3.2 (a) the ratio:

$$\frac{nnz(\tilde{A}_\tau)}{nnz(\tilde{A})},$$

when the threshold parameter τ is varied for $\tilde{A}$ constructed using two levels of neighbours. In Figure 3.2 (b), we depict the ratio:

$$\frac{nnz(L_\tau) + nnz(D_\tau)}{nnz(L) + nnz(D)},$$

where $\tilde{A}_\tau = L_\tau D_\tau L_\tau^T$ and $\tilde{A} = LDL^T$ when the threshold τ varies. We observe that the number of nonzero entries decreases more quickly in the approximations $\tilde{A}_\tau$ than in their corresponding factors. Unfortunately as we will see in the next section the numerical behaviour of the resulting preconditioner also quickly deteriorates.



(a) Ratio of the number of nonzeros in the approximation of A

(b) Ratio of the number of nonzeros in the factors of the approximation of A

Figure 3.2: M_{Mumps} - Effect on the number of nonzero entries in $\tilde{A}_\tau$ and its factors when τ is varied.

3.2.4 Numerical experiments

In Table 3.5 we depict the number of matrix-vector products observed with the M_{Frob} and M_{Mumps} preconditioners. Both full-GMRES and restarted GMRES are considered in these experiments. For the case of full-GMRES, the preconditioner M_{Mumps} reduces the number of matrix-vector products by 20% on average for the smaller sizes of each geometry. The gain tends to decrease as the size of the problem increases. The exception is for the Aircraft where we obtain a reduction of 50% whatever the size. For the case of restarted GMRES, the reduction is larger on the smaller problems (until 30%) but again shrinks when the problem becomes larger. The Aircraft case remains an exception with a reduction of around 75% on the first example; convergence is achieved on the second example with M_{Mumps} while it is not observed with M_{Frob}.

Moving from one level to two level neighbours does not introduce a significant reduction in the number of iterations as illustrated in Table 3.6. This reduction is not significant enough to balance the large extra cost in memory. The idea of using $\tilde{A}_\tau$ instead of $\tilde{A}$ to reduce the memory constraint leads to very poor preconditioners. The behaviour of this approach is illustrated in Figure 3.3. We see that the number of full-GMRES iterations using $\tilde{A}_\tau$ quickly deteriorates when entries are discarded.

The main drawback of the M_{Mumps} approach is the memory required by the factorization which becomes prohibitive for very large computations. For this reason we do not further consider this preconditioning approach in the rest of the manuscript. We will rather focus on techniques that exploit M_{Frob} in various ways to design robust preconditioners and numerical schemes.

Aircraft				
	full-GMRES		GMRES(120)	
Size	M_{Frob}	M_{Mumps}	M_{Frob}	M_{Mumps}
94 704	746	367	1 956	507
213 084	$973^{(16)}$	$508^{(16)}$	$+2000^{(16)}$	$951^{(16)}$
Almond				
	full-GMRES		GMRES(120)	
Size	M_{Frob}	M_{Mumps}	M_{Frob}	M_{Mumps}
104 793	234	202	253	215
419 172	$413^{(20)}$	$377^{(20)}$	$571^{(20)}$	$464^{(20)}$
Cetaf				
	full-GMRES		GMRES(120)	
Size	M_{Frob}	M_{Mumps}	M_{Frob}	M_{Mumps}
86 256	656	542	1 546	1 080
134 775	618	517	1 125	852
264 156	710	590	1 373	1 003
Cobra				
	full-GMRES		GMRES(120)	
Size	M_{Frob}	M_{Mumps}	M_{Frob}	M_{Mumps}
60 695	369	286	516	341
179 460	353	315	406	355

Table 3.5: Number of matrix-vector products using M_{Frob} and M_{Mumps} preconditioners. All these results are obtained on 8 processors except those marked with (k), that were run on k processors.

	one level	two level
Aircraft 23 676	160	120
Cobra 60 695	286	272
Cetaf 86 256	542	464

Table 3.6: Number of full-GMRES iterations using M_{Mumps} based on one and two levels of neighbours to construct $\tilde{A}$.

3.3 Stationary scheme as preconditioner

3.3.1 Governing ideas and properties

Any preconditioner can be used to design a stationary iterative scheme and reciprocally, any stationary iterative scheme can lead to the definition of a preconditioner. In this section we exploit this close relationship between preconditioners and stationary iterative schemes to design a new preconditioner M. This new preconditioner will eventually be based on M_{Frob} for our experiments in electromagnetics.

Let us consider the stationary scheme based on M that is given by:

$$\begin{aligned} x^{(j)} &= x^{(j-1)} + \omega M(b - Ax^{(j-1)}) \\ &= (I - \omega MA)x^{(j-1)} + \omega Mb, \end{aligned} \tag{3.5}$$

where ω is a relaxation parameter that must be set such that the spectral radius $\rho(I - \omega MA)$ is less than 1 to ensure the convergence of the scheme for any choice of $x^{(0)}$. If $e^{(j)} = x^{(j)} - x^*$

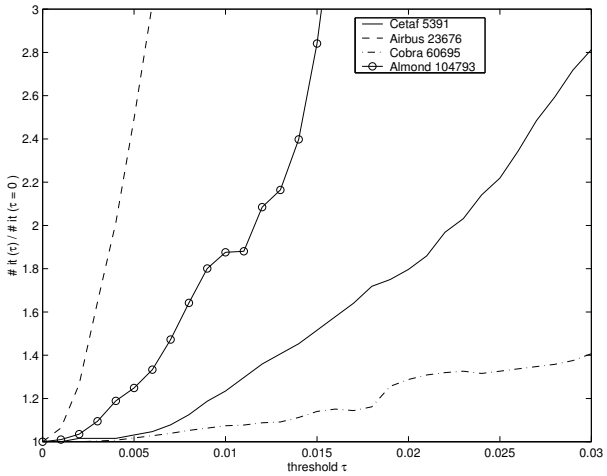


Figure 3.3: Sensitivity of the threshold τ on the degradation of the number of full-GMRES iterations. The "ratio" is the number of iterations using $\tilde{A}_\tau$ divided by the number of iterations using $\tilde{A}$ to build the preconditioner M_{Mumps}.

denotes the error associated with the j^{th} iterate $x^{(j)}$, the error equation associated with (3.5) is:

$$e^{(j)} = (I - \omega MA)e^{(j-1)}. \tag{3.6}$$

Let us now consider the stationary scheme where each iteration comprises μ steps of the scheme (3.5). The error equation associated with this new scheme can be written:

$$\tilde{e}^{(j)} \quad = \quad (I - \omega MA)^\mu \tilde{e}^{(j-1)},$$

where $\tilde{e}^{(j)} = e^{(j-\mu)}$. This latter equation can be written in the general form similar to Equation (3.6) which gives:

$$\tilde{e}^{(j)} = (I - M_{Iter(\mu)}A)\tilde{e}^{(j-1)} = (I - \omega MA)^\mu \tilde{e}^{(j-1)},$$

where $M_{Iter(\mu)} = (I - (I - \omega MA)^\mu)A^{-1}$. The matrix $M_{Iter(\mu)}$ can also be used as a preconditioner for a Krylov method. The spectrum of the preconditioned matrix is given by the following proposition.

Proposition 3.3.1 *Let $M_{Iter(\mu)}$ be the operator $(I - (I - \omega MA)^\mu)A^{-1}$. Let $\{\lambda_i\}_{i\in 1,..,n}$ be the eigenvalues of MA (or AM). The eigenvalues of the matrix $M_{Iter(\mu)}A$ (or $AM_{Iter(\mu)}$) are:*

$$\eta_i = 1 - (1 - \omega\lambda_i)^\mu.$$

Proof
This is a direct consequence of the property of the eigenvalues of polynomials of matrices. Let $p(x)$ be a polynomial. If λ is an eigenvalue of the matrix B, $p(\lambda)$ is an eigenvalue of the matrix $p(B)$.

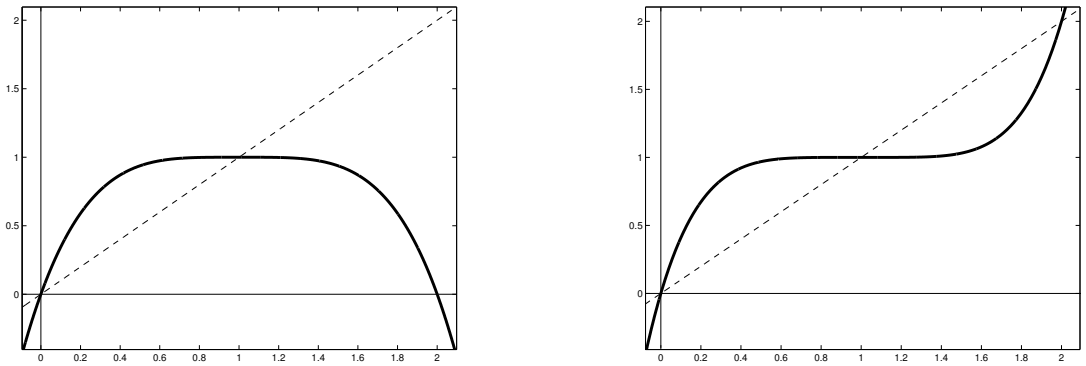
■

We focus our attention on the eigenvalue transformation produced by the $M_{Iter(\mu)}$ preconditioner. The parameter ω scales the eigenvalues of MA. For symmetric positive definite (SPD) matrices where all the eigenvalues are real positive, the function $f(y) = 1 - (1 - y)^\mu$ gives the

transformation of the eigenvalues of the preconditioned matrix $M_{Iter(\mu)}A$ (that are also those of $AM_{Iter(\mu)}$). We plot in Figure 3.4 the function $f(\lambda)$ when all the eigenvalues λ are strictly positive. For μ odd, the preconditioned matrix remains similar to a SPD matrix for any choice of ω. For μ even, this is no longer the case. Choosing ω so that $0 < \omega < \frac{2}{|\lambda_{max}|}$, forces all the eigenvalues to stay between 0 and 2 and then the preconditioned matrix remains similar to a SPD matrix. In the neighbourhood of zero, the smallest eigenvalues are moved away from the origin. The neighbourhood of one tends to remain unchanged. For μ even, the relaxation parameter prevents some eigenvalues in the neighbourhood of two becoming negative or too close to zero. By taking for example: $\omega = \frac{3}{2}|\lambda_{\max}|^{-1}$, we restrict the function to its most contracting part $]0; \frac{3}{2}]$. Large values of μ result in enlarging the interval around one where the eigenvalues are moved very close to one. An extension to the complex nonsymmetric case is possible but the transformation is no longer simple to depict.

All the eigenvalues lying in the open disk of radius one centered in (1,0) are contracted towards (1,0), those out of this disk are moved away from (1,0).

The algorithm for the construction of $M_{Iter(\mu)}$ is displayed in Algorithm 5, which of course does not require access to A^{-1}. The algorithm takes as input the residual r we want to precondition, and returns as output the preconditioned vector z after μ steps of a stationary scheme involving M.



(a) μ even (=4) (b) μ odd (=5)

Figure 3.4: Shape of the polynomial that governs the eigenvalue distribution of $M_{Iter(\mu)}$.

Algorithm 5 Stationary scheme as preconditioner: $z = M_{Iter(\mu)}r$.

/ $M_{Iter(\mu)}$ is implicitly defined by μ steps of a stationary scheme */*

1: /* Initialization */
   $z^{(0)} = 0$;
2: /* Perform μ steps of the stationary scheme */
3: **for** $\ell = 1, \cdots, \mu$ **do**
4: $z^{(\ell)} = z^{(\ell-1)} + \omega M(r - Az^{(\ell-1)})$;
5: **end for**
6: $z = z^{(\mu)}$;

3.3.2 Numerical experiments

We investigate the numerical behaviour of $M_{Iter(\mu)}$ in the electromagnetic application. In that framework, we use an accurate FMM calculation for the matrix-vector product required by GMRES

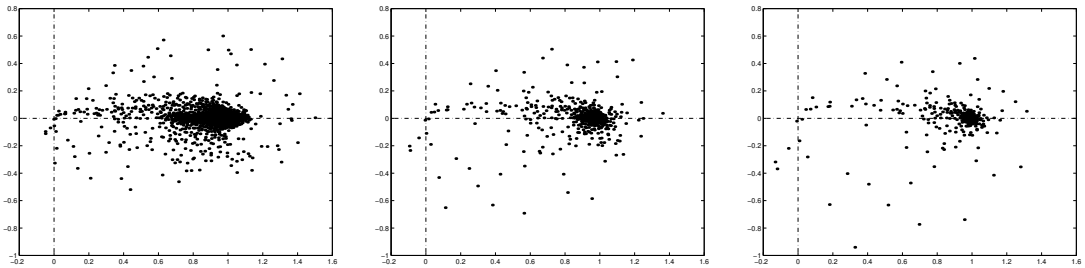
and a less accurate, cheaper to compute, FMM calculation for the matrix-vector product involved in the preconditioning operation. On average one accurate FMM takes from 1.5 to 2 times more than a less accurate FMM. This behaviour can be observed in Table 3.7 where we list the set of test examples we have considered.

Aircraft		
Size	Inner FMM	Outer FMM
94 704	3.5	4.8
213 084	6.9	11.8
591 900	17.4	30.2
Almond		
Size	Inner FMM	Outer FMM
104 793	1.8	3.0
419 172	6.2	11.2
943 137	14.3	25.5
Cetaf		
Size	Inner FMM	Outer FMM
86 256	1.9	3.7
134 775	3.2	5.2
264 159	5.6	11.4
539 100	11.5	19.3
Cobra		
Size	Inner FMM	Outer FMM
60 695	1.1	2.0
179 460	3.2	5.6

Table 3.7: Average elapsed time in seconds observed on 8 processors for a matrix-vector product using either an accurate (Outer) or a fast (Inner) FMM.

For the numerical experiments, the relaxation parameter ω in Algorithm 5 is set to $\frac{3}{2}|\lambda_{\max}|^{-1}$, where $|\lambda_{\max}|$ is estimated as the Ritz value of largest magnitude obtained from a few steps of GMRES performed in a preprocessing phase.

In Figure 4.8, we plot the spectrum of $AM_{Iter(1)}$, $AM_{Iter(2)}$ and $AM_{Iter(3)}$ for the Cetaf example of size 5 391, with $M = M_{Frob}$. As can be expected, the contraction of the neighbourhood of one to one is stronger as μ increases. The size of the cluster centered at one is much smaller for $\mu = 3$ than for $\mu = 1$. The positive part of the neighbourhood of zero is attracted towards one, and the negative part tends to move away in the left plane. For the numerical experiments,



(a) $\mu = 1$ (b) $\mu = 2$ (c) $\mu = 3$

Figure 3.5: Spectrum of $AM_{Iter(\mu)}$ for different values of μ on the Cetaf 5391.

we consider only two values for μ: $\mu = 2$ and $\mu = 3$. The case $\mu = 1$ reduces to M_{Frob} scaled by ω and is consequently not interesting. In Table 3.8 we report on the elapsed time and the number of accurate FMM calculations (i.e. number of accurate matrix-vector products) required to solve the various linear systems using GMRES(120) for the Cetaf and the Aircraft problems and GMRES(60) for the Cobra and Almond problems. We consider M_{Frob}, $M_{Iter(2)}$ and $M_{Iter(3)}$ on the four geometries varying their size. The $M_{Iter(2)}$ preconditioner requires 1.8 to 2.8 times less accurate matrix-vector products than M_{Frob}, except for the largest Almond case where a factor of 4.3 is obtained. Moreover, convergence can be achieved on the Aircraft of size 213 084 and the Almond of size 419 172 with $M_{Iter(2)}$, whereas it is not the case with M_{Frob}.

Increasing the value of μ to 3 enables us to reduce the number of accurate matrix-vector products again. We obtain a gain ranging from 1.5 to 1.9 using $M_{Iter(3)}$ in comparison with $M_{Iter(2)}$. The gain in elapsed time is less significant because applying $M_{Iter(\mu)}$ requires μ extra less accurate FMM calculations. The observed ratio in elapsed time between M_{Frob} and $M_{Iter(2)}$ varies between 1.1 and 1.5. The gains in time introduced by $M_{Iter(3)}$ range from 1.1 to 1.7 compared to M_{Frob}. The preconditioner $M_{Iter(3)}$ outperforms $M_{Iter(2)}$ both in the number of accurate FMM calculations and in the solution time. Using a larger value for μ would probably decrease the number of accurate FMM calculations, but the cost per iteration would increase so that the overall elapsed time will not reduce much. Even though we do not report any numerical experiments to assess this claim we believe that large values of μ might be an alternative to M_{Frob} when this latter preconditioner does not enable convergence.

3.4 Embedded solution scheme

For the sake of comparison with the results of the previous section, we briefly summarize below some of the results presented in [20]. In [20], the authors describe an embedded iteration scheme for the solution of large electromagnetic problems using an FMM approximation for the matrix-vector products. The idea consists in using for each outer iteration a variable preconditioner defined by a few steps of an inner Krylov method. The efficiency of such an embedded scheme relies on two main factors: the inner solver has to be preconditioned so that the residual in the inner iterations can be significantly reduced in a few steps, and the matrix-vector product within the inner and the outer solvers can be carried out with a different accuracy and thus a different computational cost. An accurate FMM is implemented in the outer solver because it governs the final accuracy of the computed solution. A less accurate FMM is used in the inner solver as it is a preconditioner for the outer scheme. The outer solver is the Flexible GMRES algorithm [32], denoted by FGMRES(m), and the inner solver is p iterations of preconditioned full-GMRES (that can also be viewed as one complete restart of GMRES(p)). The preconditioner for the inner solver is the M_{Frob} preconditioner. For the same restart value m, the storage requirement for the FGMRES algorithm is twice that for the standard GMRES algorithm, because it also stores all the variable preconditioned vectors obtained from the inner solver (see Section 2.3). The authors compare the FGMRES(m)/GMRES(p) approach with the GMRES(2*m+p) approach in order to use the same amount of memory. On medium cases (until 200 000 unknowns), FGMRES(m)/GMRES(p) already gives better performance in terms of elapsed time than GMRES(2*m+p). In Table 3.9 we report the results of [20] on the largest problems where FGMRES(30)/GMRES(60) is compared with GMRES(120). For the embedded scheme we report on the number of outer steps. The corresponding cumulated number of inner matrix-vector products are displayed after the symbol "+" in the column entitled "Mat.V". It can be seen that the combination FGMRES/GMRES with M_{Frob} leads to a remarkably robust numerical scheme. The increase in the number of outer matrix-vector products is fairly modest except on the largest aircraft test case. Thanks to this inner-outer scheme, convergence can be achieved on challenging problems where classical restarted GMRES does not converge and where full GMRES exceeds the memory of the computer. This scheme is currently the default in the AS_ELFIP code and we notice that some of the results presented in Table 3.8

Aircraft						
Size	GMRES(120)					
	M_{Frob}		$M_{Iter(2)}$		$M_{Iter(3)}$	
	# Mat.V	Time	# Mat.V	Time	# Mat.V	Time
94 704	1956	3h 13mn	692	2h 15	430	2h
213 084	*+2000*	*> 7h 56mn*	1150	8h 20	609	5h 40
591 900	*+2000*	*> 1d 57mn*	*+2000*	*> 2d*	*+2000*	*> 2d*

Almond						
Size	GMRES(60)					
	M_{Frob}		$M_{Iter(2)}$		$M_{Iter(3)}$	
	# Mat.V	Time	# Mat.V	Time	# Mat.V	Time
104 793	302	19mn	157	17mn	105	15mn
419 172	*+2000*	*> 10h*	348	2h	204	1h 45
943 137	1633	6h 36mn$^{(16)}$	377	6h	225	4h 45

Cetaf						
Size	GMRES(120)					
	M_{Frob}		$M_{Iter(2)}$		$M_{Iter(3)}$	
	# Mat.V	Time	# Mat.V	Time	# Mat.V	Time
86 256	1546	1h 43mn	628	1h 15	392	1h
134 775	1125	1h 55mn	494	1h 30	330	1h 20
264 156	1313	4h 46mn	572	3h 30	363	3h
531 900	1717	14h 08mn	708	10h	449	9h

Cobra						
Size	GMRES(60)					
	M_{Frob}		$M_{Iter(2)}$		$M_{Iter(3)}$	
	# Mat.V	Time	# Mat.V	Time	# Mat.V	Time
60 695	708	29mn	322	21mn	221	19mn
179 460	433	48mn	237	46mn	171	45mn

Table 3.8: Number of accurate matrix-vector products and elapsed time required to converge, using M_{Frob}, $M_{Iter(2)}$ and $M_{Iter(3)}$ preconditioners with the restarted GMRES method. The tests were run on 8 processors of the Compaq machine, except those marked with $^{(k)}$, that were run on k processors.

compare favourably with those presented in Table 3.9.

3.5 Conclusions

In this first part, we have presented the scattering problem arising from electromagnetism calculations. The fast multipole method allows us to consider very large problems using approximate matrix-vector products while reducing the memory requirement. The Chapter 2 describes many variants of the GMRES method, which turn out to be effective when they are combined with the Frobenius-norm minimization preconditioner M_{Frob}. But the numerical efficiency of M_{Frob} decreases as the size of the problem increases. Playing on the parameters that govern the construction of M_{Frob} to improve it often results in strongly increasing the cost of applying it without a significant effect on the number of iterations. Different techniques involving M_{Frob} are successfully investigated to improve its robustness. In the next part, we explore another strategy based on a spectral update of M_{Frob}, which could be used to complement these existing techniques.

Aircraft				
	GMRES(120)		FGMRES(30)/GMRES(60)	
	# Mat.V	Time	# Mat.V	Time
94 704	1956	3h 13mn	27+1560	2h 14mn
213 084	*+2000*	*> 7h 56mn*	34+1920	5h
591 900	*+2000*	*> 1d 57mn*	57+3300	1d 9h 45mn
Almond				
	GMRES(60)		FGMRES(15)/GMRES(30)	
	# Mat.V	Time	# Mat.V	Time
104 793	302	19mn	11+300	14mn
419 172	*+2000*	*> 10h*	20+540	1h 24mn
943 137	1633	6h 36mn$^{(16)}$	22+600	3h 32mn
Cetaf				
	GMRES(120)		FGMRES(30)/GMRES(60)	
	# Mat.V	Time	# Mat.V	Time
86 256	1546	1h 43mn	17+ 960	55mn
134 775	1125	1h 55mn	15+ 840	1h 19mn
264 156	1373	4h 46mn	17+ 960	2h 22mn
531 900	1717	14h 8mn	19+1080	6h
Cobra				
	GMRES(60)		FGMRES(15)/GMRES(30)	
	# Mat.V	Time	# Mat.V	Time
60 695	708	29mn	24+660	18mn
179 460	433	48mn	20+540	42mn

Table 3.9: Number of accurate matrix-vector products and elapsed time required to converge, using M_{Frob} with the restarted GMRES method and M_{Frob} with the inner GMRES solver of the restarted Flexible-GMRES method. The tests were run on 8 processors of the Compaq machine, except those marked with $^{(k)}$, that were run on k processors.

II

Chapter 4

Preconditioners based on a spectral low-rank update

4.1 Motivation

It is well known that the convergence of Krylov methods for solving the linear system $Ax = b$ depends to a large degree on the eigenvalue distribution; there are exceptions as the right-hand side might also play an important role and, even with a good eigenvalue distribution, the convergence can be poor [5, 42]. Additionally, quite frequently there are small eigenvalues that adversely affect the convergence. In the symmetric positive definite (SPD) case, this can be illustrated by the bound on the rate of convergence of the Conjugate Gradient method (CG) given by [40] viz.

$$||e^{(k)}||_A \leq 2 \cdot \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^k ||e^{(0)}||_A,$$

where $e^{(k)} = x^* - x^{(k)}$ denotes the error associated with the iterate at step k and:

$$\kappa(A) = \frac{\lambda_{max}}{\lambda_{min}},$$

denotes the condition number. From this bound, it can be seen that increasing the size of the smallest eigenvalues might improve the convergence rate of CG. Consequently if the smallest eigenvalues of A could be somehow "removed", the convergence of CG might be improved [50, 59, 60, 66]. Similar arguments exist for unsymmetric systems to mitigate the bad effect of the smallest eigenvalues on the rate of convergence of the unsymmetric Krylov solver [7, 28, 29, 61]. The main argument is that the Krylov methods build a polynomial expansion that should be equal to one when the argument is zero and whose roots are the eigenvalues. To get fast convergence it is necessary to find a low order polynomial with these properties (for example, strategies have been developed to improve the convergence of GMRES [76]). Clearly the presence of eigenvalues close to the origin makes this difficult.

For GMRES there are essentially two different approaches for exploiting information related to the smallest eigenvalues, that consist either in modifying the Krylov solver to derive a variant of GMRES or in using this information to design preconditioners. The first idea is to compute a few, k say, approximate eigenvectors of A corresponding to the k smallest eigenvalues in magnitude, and augment the Krylov subspace with those directions. At each restart, let $u_1, u_2, ..., u_k$ be approximate eigenvectors corresponding to the approximate eigenvalues of A closest to the origin. The updated solution of the linear system in the next restart of GMRES is extracted from $Span\{r_0, Ar_0, A^2r_0, A^3r_0, ..., A^{m-k-1}r_0, u_1, u_2, ..., u_k\}$. This approach is referred to as the *augmented subspace approach* (see [15, 61, 63, 64, 73]). The standard implementation of the restarted

GMRES algorithm is based on the Arnoldi process, and spectral information on A might be recovered during the iterations. We notice that the GMRES-DR method described in Section 2.2 belongs to this class of techniques. The second idea also exploits spectral information gathered during the Arnoldi process to determine an approximation of an invariant subspace of A associated with the eigenvalues nearest the origin, but it uses this information to construct a preconditioner or to adapt an existing one. The idea of using exact invariant subspaces to improve the eigenvalue distribution was proposed in [71]. Information from the invariant subspace associated with the smallest eigenvalues and its orthogonal complement are used to construct an adaptive preconditioner in the approach proposed in [7]. This information can be obtained from the Arnoldi decomposition of a matrix A of size n that has the form

$$AQ_m = Q_m H_m + h_{m+1,m} q_{m+1} e_m^T.$$

Let the matrix $Q_k \in \mathbb{C}^{n\times k}$ consists of the first k columns $q_1, q_2, ..., q_k$ of Q_m, and let the columns of the orthogonal matrix P_{n-k} span the orthogonal complement of $Span\{q_1, q_2, ..., q_k\}$. As $P_{n-k}^H P_{n-k} = I_{n-k}$, the columns of the matrix $[Q_k\, P_{n-k}]$ form an orthogonal basis of $\mathbb{C}^n$. In [7] the inverse of the matrix

$$M = Q_k H_k Q_k^H + P_{n-k} P_{n-k}^H$$

is used as a preconditioner. It can be expressed as:

$$M^{-1} = Q_k H_k^{-1} Q_k^H + P_{n-k} P_{n-k}^H.$$

At each restart, the preconditioner is updated by extracting new eigenvalues which are the smallest in magnitude as well as their associated eigenvectors. The algorithm proposed uses the recursion formulae of the implicitly restarted Arnoldi (IRA) method described in [83], and the determination of the preconditioner does not require the evaluation of any matrix-vector products with the matrix A in addition to those needed for the Arnoldi process. Another similar adaptive procedure to determine a preconditioner during GMRES iterations was introduced earlier in [29]. It is based on the same idea of estimating an invariant subspace corresponding to the smallest eigenvalues. The preconditioner is based on a deflation technique such that the linear system is solved exactly in an invariant subspace of dimension k corresponding to the k smallest eigenvalues of A.

Most of these schemes are combined with the GMRES procedure as they derive information directly from its internal Arnoldi process. In this part, we consider an additional explicit eigencomputation that is used to update the selected preconditioner. This makes the preconditioner independent of the Krylov solver used for the actual solution of the linear system. This extra cost can be overcome if the same linear system has to be solved for several right-hand sides, because the number of Krylov iterations can be significantly reduced. Such a situation exists for instance in radar cross section calculations in electromagnetism and is further discussed in Section 4.3.3.

In this chapter we consider two different approaches to exploit this spectral information to design a preconditioner. These techniques are inspired by schemes developed for the solution of partial differential equations such as coarse space correction in domain decomposition techniques [14, 81] or multigrid techniques [46]. The first preconditioning technique referred to as spectral low-rank update was introduced in [18]. The additive two-level spectral preconditioner was presented in [21]. This chapter is organized as follows. The main features of these two spectral preconditioners are recalled in Section 4.2. In Section 4.3, we illustrate the effect of the size of the spectral low-rank update on the convergence rate of the GMRES solver for one right-hand side. We show the gain for a complete radar cross section on the geometries described in Section 1.3.2. Since the spectral low-rank update compensates for some possible weaknesses of the approximate inverse, we illustrate that a balance has to be found between the two components of the resulting preconditioner. Then, we show that the gain induced by the low-rank update becomes larger as the size of the restart in GMRES decreases. A possible combination of this preconditioner with the Flexible-GMRES method is also investigated. Finally, since many right-hand sides have to be

solved, we illustrate the benefit of using such a preconditioner in the context of seed GMRES. In Section 4.3.8, we conclude with some experiments with the additive two-level approach.

4.2 Description of the spectral preconditioners

Many preconditioners are able to cluster most of the eigenvalues close to one but still leave a few close to the origin. To move these eigenvalues close to one might be possible by tuning the parameters that control these preconditioners; however this is often difficult and might lead to very expensive preconditioners to setup and to apply. We refer to Table 4.8 in Section 4.3.4 for an illustration of this fact in electromagnetism calculations. However, other alternatives can be considered. In the next sections we describe two techniques that attempt to improve a prescribed preconditioner using spectral information on the preconditioned matrix.

4.2.1 Spectral low-rank update preconditioner

In [18] a spectral low-rank update (SLRU) preconditioning technique that exploits the knowledge of the eigenvectors associated with the smallest eigenvalues is described. The proposed preconditioners shift the corresponding eigenvalues close to one and numerical examples show the relevance of this approach to speed up the convergence of Krylov solvers. Roughly speaking, the proposed technique consists in solving exactly the preconditioned system in the low dimensional space spanned by its eigenvectors associated with the eigenvalues closest to the origin. This is then used to update the preconditioned residual. Let us now briefly recall the formulation of this preconditioner as well as its spectral properties. We notice that its formulation depends on the spectral information available, that is whether the eigenvectors of AM or MA are available.

Let us consider the solution of the linear system

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

where A is a nonsingular matrix in $\mathbb{C}^{n\times n}$, and x and b are vectors of $\mathbb{C}^n$. The linear system is solved using a preconditioned Krylov solver and we denote by M the initial preconditioner. Let $\{\lambda_1, \cdots, \lambda_n\}$ be the set of eigenvalues of MA where the multiple eigenvalues are repeated. Let the columns of U be the basis of a right invariant subspace of MA of dimension k. Suppose without loss of generality that $MAU = UJ_k$ where the eigenvalues of J_k are $\{\lambda_1, \cdots, \lambda_k\}$. Using this information we can design a preconditioner for the linear system for which we can derive the following result related to the spectrum of the preconditioned matrix.

Proposition 4.2.1 *Let W be such that $A_c = W^H AU$ is nonsingular, $M_c = UA_c^{-1}W^H$ and*

$$M_{SLRU(MA,k)} = M + M_c. \tag{4.2}$$

Then the eigenvalues η_i of $M_{SLRU(MA,k)}A$ (or $AM_{SLRU(MA,k)}$) are

$$\begin{cases} \eta_i = 1 + \lambda_i & \text{if} \quad i \le k, \\ \eta_i = \lambda_i & \text{if} \quad i > k. \end{cases}$$

Proof
Let $U^\perp$ be an orthogonal complement of U. We have

$$MA\left[UU^\perp\right] = \left[UU^\perp\right]\begin{pmatrix} J_k & E \\ 0 & F \end{pmatrix},$$

where the eigenvalues of J_k are $\{\lambda_1, \cdots, \lambda_k\}$ and those of F are $\{\lambda_{k+1}, \cdots, \lambda_n\}$. Consequently we have

$$(M + M_c)A\left[UU^\perp\right] = \left[UU^\perp\right]\begin{pmatrix} I + J_k & E + A_c^{-1}W^H AU^\perp \\ 0 & F \end{pmatrix},$$

which concludes the proof.

■

Notice that the result above is similar to [18, Prop. 2] but we no longer require the assumption that MA is diagonalizable. This result indicates that the preconditioner $M_{SLRU(MA,k)}$ can be used either as a right or as a left preconditioner. In both cases, the preconditioned matrix has the same spectrum.

Similarly, assume that a basis V of a right invariant subspace of AM (and no longer of MA) associated with $\{\lambda_1, \cdots, \lambda_k\}$ is given. Then, MV is a basis of the right invariant subspace of MA associated with the same eigenvalues and the following property holds, by setting $U = MV$ in Proposition 4.2.1.

Proposition 4.2.2 *Let W be such that $A_c = W^H AMV$ is nonsingular, $M_c = MVA_c^{-1}W^H$ and $M_{SLRU(AM,k)} = M + M_c$. The eigenvalues η_i of $AM_{SLRU(AM,k)}$ (or $M_{SLRU(AM,k)}A$) are*

$$\begin{cases} \eta_i = 1 + \lambda_i & \text{if} \quad i \leq k, \\ \eta_i = \lambda_i & \text{if} \quad i > k. \end{cases}$$

We notice again that this result is similar to [18, Prop. 4] but it relies on weaker assumptions because it does not require that AM is diagonalizable.

The matrix M_c is the projection of the preconditioned matrix on the space spanned by the eigenvectors of the preconditioned matrix associated with the smallest eigenvalues in magnitude. It is a rank-k correction of M, which ensures that the new preconditioned system no longer has eigenvalues with magnitude smaller than $|\lambda_{k+1}|$. We made the assumption that the prescribed preconditioner was such that the preconditioned matrix already has most of its eigenvalues close to one, so that shifting the remaining smallest eigenvalues close to one makes sense. Similar preconditioners can be derived if other shifts are preferred and/or other parts of the spectrum are identified as responsible for slowing down the convergence. In practice, the columns of V are the right eigenvectors associated with the k smallest eigenvalues of the preconditioned matrix and $W = V$ is often selected. Finally, for the sake of simplicity of the notation, $M_{SLRU(AM,k)}$ and $M_{SLRU(MA,k)}$ will be denoted by $M_{SLRU(k)}$ in the rest of this manuscript. Whether it is built from the eigenvector of MA or AM will be clear in the context.

4.2.2 Additive spectral two-level preconditioner

Other numerical techniques can be designed to take advantage of the spectral information available from the preconditioned matrix. From the philosophy of the two-grid approach solving elliptic partial differential equations, we can view the prescribed preconditioner as a smoother. The eigenvectors associated with the "smooth" modes of the preconditioned system define the coarse space of a two-grid scheme. In classical multigrid, the coarse space is not defined explicitly through the knowledge of the eigencomponents but by the selection of a space that is expected to capture them. In our framework the coarse space is defined by the columns of U which are a basis of the invariant subspace of dimension k associated with the smallest eigenvalues in magnitude of MA (i.e. the components of the error that are not efficiently damped by the smoother that corresponds to the slowly converging modes). In that context, the prolongation operator is $P = U$, the restriction operator is denoted by $R = W^H$ and the matrix involved in the coarse grid error problem is defined by a Galerkin formula $A_c = RAP$. Multiplicative and additive two-grid techniques are described in [21] that both enable us to define preconditioned systems that have the same spectrum. The latter variant, the additive approach, has some computation advantages since it exhibits very similar efficiency in reducing the number of iterations of the Krylov solvers [21]. We describe below the additive variant and establish its spectral properties using again weaker assumptions than those considered in [21].

In the additive algorithm, the coarse grid correction and the smoothing operation are decoupled. Each process generates an approximation of the preconditioned residual vector in complementary subspaces. The coarse grid correction computes only components in the space spanned by the eigenvectors associated with the few selected small eigenvalues, while at the end of the smoothing step the preconditioned residual is filtered so that only components in the complementary subspace are retained. These two contributions are summed together for the solution update. A simple additive two-level multigrid is illustrated in Algorithm 6. In this algorithm we follow [30] and select the procedure advocated in [86]; we define the filtering operators using the grid transfer operators as $(I - UW^H)$. This operator is supposed to remove all the components in the U directions. A natural choice is to select W^H so that $(I - UW^H)U = 0$ (i.e. $W^H U = I$).

Algorithm 6 Additive spectral preconditioner

set $z^1 = 0$;
for $\ell = 1, \ldots, iter$ **do**
  Compute the residual: $s^\ell = r - Az^\ell$;
  / Compute the high and low frequency corrections */*
  / High frequency correction */*
  / Damp all the frequencies of the error */*
  $e_1^{\ell,0} = 0$;
  for $j = 1, \ldots, \mu_1 + \mu_2$ **do**
    $e_1^{\ell,j} = e_1^{\ell,j-1} + \omega M(s^\ell - Ae_1^{\ell,j-1})$;
  end for
  / Filter the high frequencies of the correction */*
  $c_1^\ell = (I - UW^H)e_1^{\ell,\mu}$;
  Low frequency correction: $c_2^\ell = UA_c^{-1}W^H s^\ell$;
  Update the solution: $z^{\ell+1} = z^\ell + c_1^\ell + c_2^\ell$;
end for
$z = z^{iter}$;

Proposition 4.2.3 *Let W be such that $A_c = W^H AU$ has full rank and satisfies $(I - UW^H)U = 0$. The preconditioning operation described in Algorithm 6 can be written in the form $z = M_{Add}\, r$. In the case $iter = 1$ the preconditioner M_{Add} has the following expression:*

$$M_{Add} = UA_c^{-1}W^H + (I - UW^H)(I - (I - \omega MA)^\mu)A^{-1}. \tag{4.3}$$

Proof
Hereby we remove the superscript ℓ in Algorithm 6, as we analyse the case $iter = 1$. The smoothing steps generate a sequence of vectors of the form $e_1^j = (I - \omega MA)e_1^{j-1} + \omega Mr$, that can be written $e_1^\mu = [I - (I - \omega MA)^\mu]A^{-1}r$ because $e_1^{\ell,0} = 0$. Consequently, $z_1 = (UA_c^{-1}W^H + (I - (I - \omega MA)^\mu]A^{-1})r$.

■

We notice that the term associated with the smoothing steps: $(I - (I - \omega MA)^\mu)A^{-1}$ is identical to the iteration matrix that defines $M_{Iter(\mu)}$ in Section 3.3 (see Proposition 3.3.1).

Proposition 4.2.4 *The preconditioner M_{Add} defined by Proposition 4.2.3 is such that the preconditioned matrix $M_{Add}A$ (or AM_{Add}) has eigenvalues:*

$$\begin{cases} \eta_i = 1 & \text{if} \quad i \le k, \\ \eta_i = 1 - (1 - \omega\lambda_i)^\mu & \text{if} \quad i > k. \end{cases}$$

Proof
Let $U^\perp$ be an orthogonal complement of U. We have

$$MA\left[UU^\perp\right] = \left[UU^\perp\right]\begin{pmatrix} J_k & E \\ 0 & F \end{pmatrix},$$

where the eigenvalues of J_k are $\{\lambda_1, \cdots, \lambda_k\}$ and those of F are $\{\lambda_{k+1}, \cdots, \lambda_n\}$.

We first show that $M_{Add}AU = U$. We have $(I-\omega MA)U = U(I-\omega J_k)$, then $(I-\omega MA)^\mu U = U(I-\omega J_k)^\mu$. Consequently we obtain:

$$M_{Add}AU = UA_c^{-1}W^HAU + (I - UW^H)U(I - \omega J_k)^\mu = U,$$

because $(I - UW^H)U = 0$ and $UA_c^{-1}W^HAU = U$.

We now show that $M_{Add}AU^\perp = \left[UU^\perp\right]\begin{pmatrix} A_c^{-1}W^HAU^\perp - W^HU^\perp(I-(I-\omega F)^\mu) \\ I-(I-\omega F)^\mu \end{pmatrix}$.

We first observe that $(I - \omega MA)U^\perp = \left[UU^\perp\right]\begin{pmatrix} -\omega E \\ (I-\omega F) \end{pmatrix}$, that generalizes to:

$$(I - \omega MA)^\mu U^\perp = \left[UU^\perp\right]\begin{pmatrix} \mathcal{P}(J_k, E, F) \\ (I-\omega F)^\mu \end{pmatrix},$$

where $\mathcal{P}$ is a polynomial.

It follows that $(I-(I-\omega MA)^\mu)U^\perp = \left[UU^\perp\right]\begin{pmatrix} \mathcal{P}(J_k, E, F) \\ I-(I-\omega F)^\mu \end{pmatrix}$, then:

$$\begin{aligned}
(I - UW^H)(I-(I-\omega MA)^\mu)U^\perp &= (I - UW^H)\left[UU^\perp\right]\begin{pmatrix} \mathcal{P}(J_k, E, F) \\ I-(I-\omega F)^\mu \end{pmatrix} \\
&= \left[0\ (I - UW^H)U^\perp\right]\begin{pmatrix} \mathcal{P}(J_k, E, F) \\ I-(I-\omega F)^\mu \end{pmatrix} \\
&= (I - UW^H)U^\perp\,(I-(I-\omega F)^\mu) \\
&= -UW^HU^\perp\,(I-(I-\omega F)^\mu) + U^\perp\,(I-(I-\omega F)^\mu).
\end{aligned}$$

Consequently, $M_{Add}AU^\perp = \left[UU^\perp\right]\begin{pmatrix} A_c^{-1}W^HAU^\perp - W^HU^\perp(I-(I-\omega F)^\mu) \\ I-(I-\omega F)^\mu \end{pmatrix}$.

It follows that:

$$M_{Add}A\left[UU^\perp\right] = \left[UU^\perp\right]\begin{pmatrix} I & A_c^{-1}W^HAU^\perp - W^HU^\perp(I-(I-\omega F)^\mu) \\ 0 & I-(I-\omega F)^\mu \end{pmatrix},$$

which completes the proof.

■

For the choice of W, we use $W = QR^{-H}$, where $V = QR$, to ensure that $W^HV = I$.

The governing ideas that led to the definition of M_{SLRU} and M_{Add} are quite different even though they exploit the same ingredients. In M_{SLRU}, we project the preconditioned matrix on U and use the solution in this low dimensional space to update the preconditioner. In M_{Add}, we project the original matrix A on the space spanned by U and solve the error equation in that low dimensional space. The update is then performed so that it does not mix with the correction computed by the smoothing iterations. This latter technique is very close to the spirit of classical two-grid methods for partial differential equations.

For $\mu = 1$, M_{Add} is fairly similar to M_{SLRU}, the difference is that M_{Add} splits the correction so that the coarse space part component does not mix with the component that comes from the prescribed preconditioner M. The result is that the k smallest eigenvalues are moved exactly to

one with M_{Add} while they are shifted by one with M_{SLRU}. This observation assumes knowledge of the exact right eigenvectors. In practice, because only the computed eigenvectors are available we can expect that the two preconditioners will have very similar numerical behaviour. For $\mu > 1$, we expect that M_{Add} outperforms M_{SLRU} as $M_{Iter(\mu)}$ outperforms M_{Frob} as illustrated in Section 3.3.

4.2.3 Some computational considerations

In this section, we investigate the computational cost in terms of floating-point operations as well as in extra memory requirements associated with the two preconditioners M_{SLRU} and M_{Add}. In Table 4.1 we summarize the main numerical kernels involved in the spectral preconditioners as well as their numerical complexity in floating-point operations. These numerical kernels are involved either in the setup phase to construct the preconditioners or in the iterative process at each step of the Krylov method. We assume that A and M are $n \times n$ matrices, V and W are $n \times k$ matrices and z is a vector of size n. We denote by $nnz(A)$ (resp. $nnz(M)$) the number of nonzero entries in A (resp. M). In that table, because $k \ll n$, we do not consider the computation that has a complexity which is polynomial in k like the LU factorization of A_c which is $\frac{2}{3}k^3$.

Basic kernel	# Flop
Mz	$2nnz(M) - n$
Setup phase	**# Flop**
$W^H AMV$	$2k\,(nnz(A) + nnz(M) + n(k-1))$
$W^H AV$	$k\,(2nnz(A) + n(2k-1))$
$V = QR$	$2nk(k-3) + 4n$
Basic kernel in M_{Add}	**# Flop**
1. $z = 0$ 2. **for** $j = 1, \ldots, \mu$ **do** 3. $\quad z = z + \omega M(b - Az)$ 4. **end for**	$(\mu - 1)\,(2nnz(A) + 2nnz(M) + n) + 2nnz(M)$
$(I - VW^H)z$	$4kn$
$VA_c^{-1}W^H z$	$(4k-1)n$
Preconditioner	**# Flop**
$M_{Add}z$	$8kn + (\mu - 1)\,(2nnz(A) + 2nnz(M) + n) + 2nnz(M)$
$M_{SLRU(AM)}z$	$2nnz(M) + (4k-1)n$
$M_{SLRU(MA)}z$	$2nnz(M) + (4k-1)n$

Table 4.1: Numerical kernels and associated floating-point cost.

We see that the setup of the coarse space operator involved in $M_{SLRU(AM)}$ (i.e. $W^H AMV$) is more expensive than the one of $M_{SLRU(MA)}$ (i.e. $W^H AV$), thanks to a product by M. The setup of M_{Add} is more expensive than that of $M_{SLRU(MA)}$ because it requires an extra QR factorization of V. During the iterations, both $M_{SLRU(AM)}$ and $M_{SLRU(MA)}$ have the same application cost. M_{Add} is more sophisticated than M_{SLRU}, and is also potentially much more expensive. To outperform M_{SLRU}, M_{Add} should significantly speed up the convergence.

From a memory point of view M_{Add} is also more demanding than M_{SLRU}. This latter requires the storage of k vectors of dimension n, the former requires slightly more than twice this amount. Both V and W^H (in its implicit form QR^{-H} for M_{Add}) have to be stored. Similarly to the floating-point complexity calculation, we assume that the storage required to store the LU factors of A_c and the R factor of the QR factorization of V involved in M_{Add} are negligible

because they are of size $\mathcal{O}(k^2)$.

4.3 Application to electromagnetic calculations

4.3.1 Presentation of an eigensolver: ARPACK

The spectral information on AM_{Frob} is computed by an external eigensolver ARPACK [58] in a preprocessing phase. The ARPACK software is a collection of Fortran77 subroutines designed to solve large scale symmetric, nonsymmetric, and generalized eigenvalue problems from significant application areas. The package is designed to compute a few eigenpairs with user specified features for the selection of the eigenvalues such as the largest real part or the largest magnitude, or the smallest real part or the smallest magnitude, etc. This software is based upon an algorithmic variant of the Arnoldi process called the Implicitly Restarted Arnoldi Method (IRAM) [57]. A matrix factorization is not required, the matrix-vector product can be defined via a reverse communication mechanism.

The numerical accuracy of the computed eigenpairs is user specified. The backward error associated with a normwise approximate eigenpair (λ, u) of a matrix A [84] is defined by:

$$\min_{\Delta A} \{\xi > 0 : \|\Delta A\| \leq \xi \|A\| : (A + \Delta A)u = \lambda u\} = \frac{\|Au - \lambda u\|}{\|A\|}. \tag{4.4}$$

Let H_j be the Hessenberg matrix from an Arnoldi procedure and y an eigenvector of H_j associated with a smallest eigenvalue λ. By applying the eigenvector y in the Arnoldi relationship, we obtain:

$$\begin{aligned} AQ_j y &= Q_j H_j y + h_{j+1,j} q_{j+1}(e_j^T y), \qquad i.e: \\ AQ_j y &= \lambda Q_j y + h_{j+1,j} q_{j+1}(e_j^T y), \qquad i.e: \\ Au - \lambda u &= (e_k^T y) h_{j+1,j} q_{j+1}, \end{aligned}$$

where $u = Q_j y$. The quantity $\|Au - \lambda u\|$ can then be estimated by $|e_k^T y|.|h_{j+1,j}|$. Since $|\lambda| \leq \|H_j\| \leq \|A\|$, an upper bound for the backward error associated with a normwise approximate eigenpair (λ, u) can be deduced:

$$\frac{\|Au - \lambda u\|}{\|A\|} \leq \frac{|e_k^T y|.|h_{j+1,j}|}{|\lambda|}.$$

The IRAM procedure from ARPACK stops at the end of the first restart that offers k Ritz value approximations $(\lambda, Q_m y)$ satisfying:

$$|e_k^T y|.|h_{j+1,j}| \leq tol.|\lambda|, \tag{4.5}$$

where tol is a user-defined tolerance [58, p 70].

ARPACK is called in sequential mode, the use of a parallel version of ARPACK (PARPACK) being impossible due to the complexity of the out-of-core configuration of the AS_ELFIP code used to perform the matrix-vector product in the reverse communication. The cost of a parallel use of ARPACK in terms of elapsed time and number of matrix-vector products is then estimated to be taken in account when we assess performance. ARPACK uses the routine ZNAUPD that implements the IRAM procedure. Once the desired Ritz values have converged, the subroutine ZNEUPD computes associated approximate Ritz vectors. These routines work in forward mode; that is they only work on A (i.e. no invert).

4.3.2 Efficiency of the preconditioner with respect to the rank of the update

In Table 4.2 we indicate, for each geometry described in Section 1.3.2, the number of unknowns we consider that are associated with each linear system, the density of M_{Frob}, the angular section considered (given in spherical coordinates, see Figure 1.5) as well as "# RHS", the number of right-hand sides to be solved for the complete monostatic calculation. This table defines the four test cases we consider in this chapter. The inital guess is set to the zero vector. In Figure 4.1, we

Geometry	Size	Density	θ	φ	# RHS
Cetaf	5 391	3.3 %	(-90) - 90	0	181
Aircraft	23 676	0.94 %	90	0 - 180	181
Cobra	60 695	0.24 %	0 - 90	0	91
Almond	104 793	0.19 %	90	0 - 180	181

Table 4.2: Angular section of interest for each geometry; the sampling step is one degree.

use the symbol "x" to plot the spectrum of AM_{Frob}, the matrix preconditioned with the Frobenius preconditioner, for the Cetaf case. As can be observed, M_{Frob} succeeds in clustering most of the eigenvalues around $(1.0, 0.0)$. Such a distribution is highly desirable to get fast convergence of Krylov solvers. Nevertheless the remaining eigenvalues nearest to zero can potentially slow down the convergence. Using the symbol "o" we plot, in Figure 4.1, the spectrum of the matrix preconditioned with $M_{SLRU(20)}$. We observe that the 20 smallest eigenvalues of the matrix AM_{Frob} have been shifted close to one, in agreement with Proposition 4.2.2. Consequently, we expect the Krylov solver to perform better with $M_{SLRU(20)}$ than with M_{Frob}. In Figure 4.2, we plot the

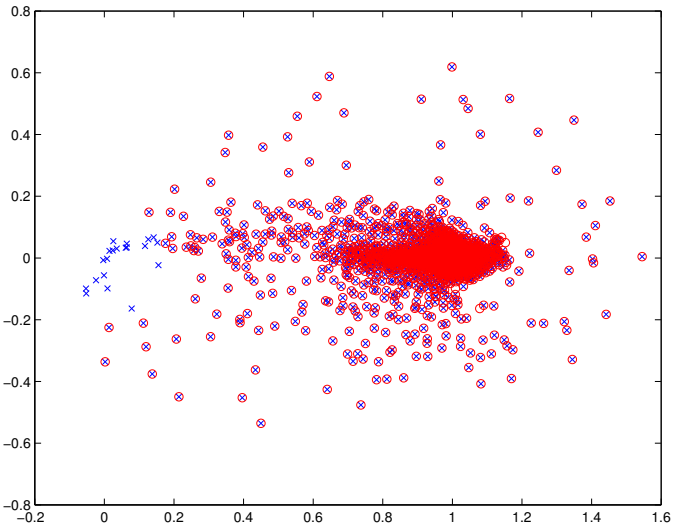


Figure 4.1: Spectrum of AM_{Frob} denoted by x and $AM_{SLRU(20)}$ depicted with o on the Cetaf test problem.

convergence histories obtained by varying the size of the low rank update. It can be observed that the larger the rank, the faster the convergence of full GMRES. However, in going from 15 to 20 the gain is negligible and going beyond 20 does not give further improvements.

As we mentioned earlier, we have used this technique for monostatic calculations. Because many linear systems with the same coefficient matrix but different right-hand sides have to be

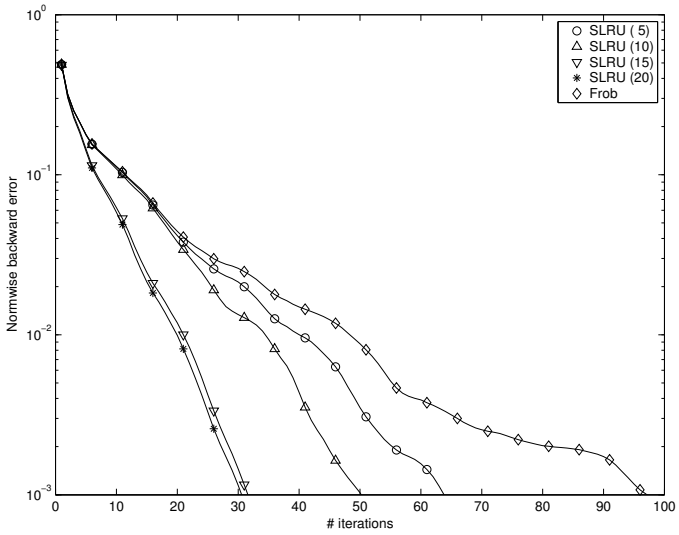


Figure 4.2: Convergence history when varying the rank of the update on the Cetaf example, using full GMRES.

solved, some are easier to solve than others. In Figure 4.3, we illustrate an important feature of the low rank update by showing the number of full GMRES iterations for convergence for a difficult right–hand side and for an easy one. It can be observed that, when the number of shifted eigenvalues increases, the number of iterations to reach convergence decreases. Furthermore, there is not much difference between a difficult right–hand side and an easy one when the rank of the update increases. Later in this chapter, we illustrate the advantage of this feature in the context of restarted GMRES and seed GMRES.

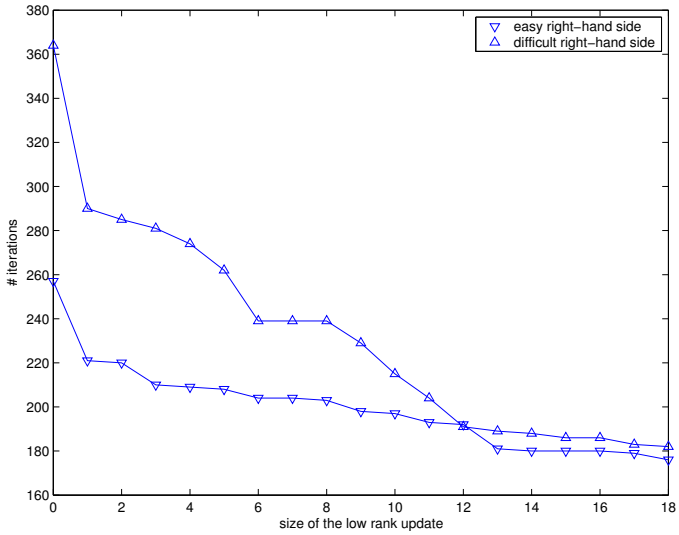


Figure 4.3: Number of full GMRES iterations when varying k in $M_{SLRU(k)}$ on the Cobra test problem.

4.3.3 Results on a complete monostatic calculation

The use of a preconditioner is often beneficial. However, its usefulness depends not only on its effect on convergence but also its construction time and the time spent in applying it at each step. In Table 4.3, we give the construction time for the FMM operator, for M_{Frob} and for $M_{SLRU(k)}$. In our experiments, the eigenvalue calculation is performed in a preprocessing phase using ARPACK, that represents the main part of the time required to setup $M_{SLRU(k)}$. In that table, we also display the average time spent in one FMM operation, one product by M_{Frob}, and one application of $M_{SLRU(k)}$ for a number of processors denoted by "# Proc".

We expect to reduce the overall number of iterations, but each new iteration is more expensive. One application of $M_{SLRU(k)}$ compared with one application of M_{Frob} introduces around $4.k.n$ additional flops, where k is the chosen number of eigenvalues and n the size of the problem. On our test examples the extra cost per iteration in elapsed time ranges from 6% (Cobra case) to 35% (Almond case), but it remains small in comparison to the FMM application times. In

			Construction Time			Application Time		
Geometry	# Proc	k	FMM	M_{Frob}	$M_{SLRU(k)}$	FMM	M_{Frob}	$M_{SLRU(k)}$
Cetaf	8	20	13 s	25 s	45 s	0.21 s	0.03 s	0.04 s
Aircraft	32	20	27 s	51 s	19 mn	0.83 s	0.12 s	0.15 s
Cobra	32	15	36 s	73 s	28 mn	1.26 s	0.11 s	0.12 s
Almond	32	60	59 s	3 mn	2 h	1.91 s	0.14 s	0.19 s

Table 4.3: Average elapsed time per matrix-vector product.

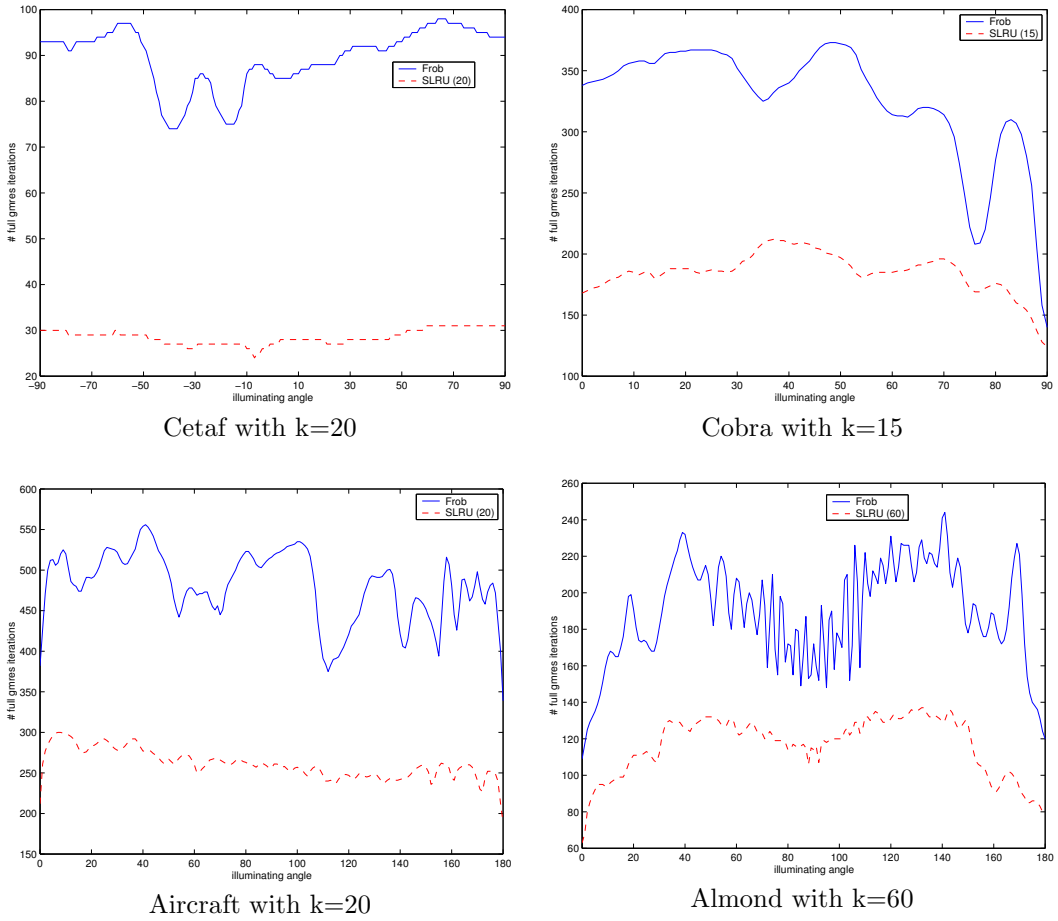
Figure 4.4, we show the number of full GMRES iterations for each right-hand side using either M_{Frob} (solid line) or $M_{SLRU(k)}$ (dashed line). For each geometry, the value of k is given in the third column. While without the spectral preconditioner, the numbers of iterations between one right–hand side and another vary a lot, with the spectral preconditioner the number of iterations per right–hand side is more similar and almost constant for some geometries. This behaviour was already observed in Figure 4.3. Table 4.4 summarizes the cumulated number of matrix–vector

		M_{Frob}		$M_{SLRU(k)}$	
Geometry	# Proc	# Mat.V	Time	# Mat.V	Time
Cetaf	8	16 391	1 h 40 mn	5 349	47 mn
Aircraft	32	87 121	46 h	47 385	18h 40 mn
Cobra	32	29 777	21 h	16 921	8h 30 mn
Almond	32	34 375	25 h 30 mn	21 273	14h 40 mn

Table 4.4: Cost for a complete monostatic calculation.

products and the total elapsed solution time for a complete radar cross section calculation for each geometry using full GMRES. For all geometries except the Cetaf, 181 linear systems are solved; only 91 are considered for the Cetaf test problem. Depending on the geometry, the overall gain ranges from a factor of 1.6 to 3 for both the CPU time and the total number of GMRES iterations. It should be pointed out that this could be improved on some examples if more eigenvalues were shifted. Our purpose in these experiments is to illustrate the potential of $M_{SLRU(k)}$. We did not try to find the best values of k for each geometry. For example, by shifting 10 more small eigenvalues for the Cobra case, we move from a factor of 1.6 to a factor of 3.

The extra cost for computing the eigenspace during the preprocessing phase in terms of matrix–vector products by AM_{Frob} as well as the corresponding elapsed time is displayed in Table 4.5. By fixing the ARPACK parameter tol defined in Equation (4.5) to 10^{-1}, the backward error of



Cetaf with k=20 Cobra with k=15

Aircraft with k=20 Almond with k=60

Figure 4.4: Number of full GMRES iterations with M_{Frob} and $M_{SLRU(k)}$ for the different incident angles for each geometry.

the eigenvectors after a construction of a Krylov space of size "# Mat.V" ranges from 10^{-7} to 10^{-6}, whatever the test case.

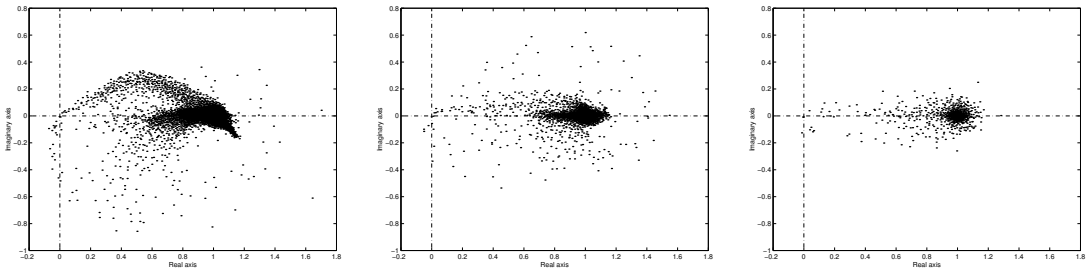
In that table, we also give the number of right-hand sides, # RHS, in the monostatic calculation for which the gain introduced by $M_{SLRU(k)}$ compensates for the cost of computing the preconditioner. It can be seen that the preprocessing calculation is quickly amortized when a few right-hand sides need to be solved.

Geometry	# Proc	# Mat.V	Time	# RHS
Cetaf	8	170	45 s	3
Aircraft	32	1 000	19 mn	6
Cobra	32	1 000	28 mn	6
Almond	32	2 200	2 h	32

Table 4.5: Cost of the eigencomputation preprocessing phase.

4.3.4 Balancing the two components of the preconditioner

Although we save a significant number of iterations using $M_{SLRU(k)}$, we might ask whether it could be more effective to use a better M_{Frob}, by allowing the preconditioner to have more nonzero entries, combined with a migration of only some smallest eigenvalues, or a worse M_{Frob}, but combined with a migration of many smallest eigenvalues. To deal with this, we first investigate the effect of the quality of M_{Frob} on the calculation time of the smallest eigenvalues of AM_{Frob}. In that respect, we vary the number of nonzeros per column leading to different densities of M_{Frob}. In Figure 4.5, we display the spectrum of AM_{Frob} on the Cetaf example for various values of the density. As we might expect, the denser M_{Frob} is, the better the clustering around one and the fewer eigenvalues close to zero; moreover these are better separated. A consequence for the eigensolver is that the few well separated eigenvalues that are near zero for the largest density of the preconditioner are more easily computed. When the density is relaxed, the eigenvalue cluster close to zero becomes wider and the eigensolver has more difficulty in computing those near zero. To illustrate this claim we show, in Table 4.6, the number of matrix–vector products and the corresponding elapsed time required by ARPACK to find the eigenvectors associated with the 60 smallest eigenvalues as the density increases. As expected, we observe that the denser the preconditioner, the easier it is for the eigensolver to find the smallest eigenvalues. A question



(a) 0.7 % (b) 3.3 % (c) 15.7 %

Figure 4.5: Spectrum of AM_{Frob} for various density of M_{Frob}.

Density	# Mat.V	Time
0.7%	365	7 mn 57 s
3.3%	240	4 mn 41 s
15.7%	152	2 mn 52 s

Table 4.6: Number of matrix–vector products and elapsed time (one processor) required to compute the 60 smallest eigenvalues of AM_{Frob} for the Cetaf when the density of M_{Frob} is varied.

that seems natural to raise is: how many eigenvalues should be shifted for these three densities of M_{Frob} to get convergence in the same number of full GMRES iterations ? On the Cetaf for a difficult angle and a density equal to 3.3 %, full GMRES needs 98 iterations to converge. Using $M_{SLRU(20)}$ GMRES needs 31 iterations to converge. The number 31 is taken as a reference value to compare the performance with the three densities. Table 4.7 shows the cost for ARPACK to compute the corresponding number of eigenvalues for each density and the total cost of computing the preconditioner.

To obtain the same number of full GMRES iterations, we need to shift more and more eigenvalues as we decrease the density. On the other hand, the construction cost of M_{Frob} with a low density is cheaper than with a higher density. There is a trade-off to be found between a cheap M_{Frob} that requires shifting many eigenvalues that might be difficult to compute, and a more

expensive M_{Frob} where only a few eigenvalues need to be shifted to get a similar convergence behaviour. As Table 4.7 shows, the medium density 3.3% offers the best trade-off among the three densities considered. The preconditioner already works well and only a few eigenvalues need to be shifted. In each case shown in Table 4.7, 31 iterations of preconditioned full GMRES are needed for convergence. The symbol "k" is the number of eigenvalues computed by ARPACK. Let us il-

M_{Frob} Construction		Eigensolver		
Density	Time	k	Time	Total
0.7%	41 s	54	7 mn 49 s	8 mn 30 s
3.3%	143 s	20	2 mn 41 s	5 mn 04 s
15.7%	1114 s	2	1 mn 04 s	19 mn 38 s

Table 4.7: Construction times when varying the M_{Frob} density on the Cetaf test problem.

lustrate, on another example, the advantage of using $M_{SLRU(k)}$ rather than increasing the density of M_{Frob}. We consider now the Almond test problem, that is the biggest, using two densities for M_{Frob}. The targeted number of iterations of full GMRES is 157 that is obtained with a density of 0.19% and by shifting 30 eigenvalues using $M_{SLRU(30)}$. To get the same number of iterations without shifting eigenvalues, we need to increase the density of M_{Frob} to 1.76 %. In Table 4.8, we show the computation cost for both preconditioners. It can be seen that the eigencalculation in the preprocessing phase of $M_{SLRU(30)}$ combined with the low cost of its M_{Frob} component is significantly less expensive than the denser M_{Frob} that exhibits the same convergence property. We compare the application times for these two approaches and the time to obtain the solution. The second approach is four times as expensive, needing 1.21 s per application as opposed to the first with 0.38 s. The use of M_{Frob} with a density of 0.19% in the first case would have cost 0.36 s; it means that $M_{SLRU(30)}$ yields an extra cost of only 0.02 s per application. Moreover, it gives a smaller solution time and a much cheaper setup time. Setup time appears to be too dominant with respect to solution time, but these results are obtained on just one angle. For solutions on an angular section, we will pay this setup time only once.

Setup					Solution		
M_{Frob}		Eigensolver		Total	Application Time		Total
Density	Time	k	Time	Time	M_{Frob}	$M_{SLRU(30)}$	Time
0.19%	6 mn	30	3 h	3 h 06 mn	-	0.38 s	8 mn 30 s
1.76%	5 h 40 mn	0	-	5 h 40 mn	1.21 s	-	10 mn 48 s

Table 4.8: Comparison of a denser M_{Frob} with a sparser $M_{SLRU(30)}$ on 8 processors on the Almond test example to obtain 157 iterations with full GMRES.

4.3.5 Sensitivity of the restarted GMRES method

All the numerical experiments reported on so far have been obtained with full GMRES. In this section we will investigate the effect of restarted GMRES on the efficiency of the preconditioners. For each value m of the restart, we show in Table 4.9 the number of GMRES iterations of M_{Frob} and $M_{SLRU(k)}$ on an easy and a hard right-hand side. The illuminating direction (θ, ϕ) corresponding to an easy right-hand side is set to $(90^o, 140^o)$ for the Aircraft case, $(-40^o, 0^o)$ for the Cetaf case, $(75^o, 0^o)$ for the Cobra case and $(90^o, 0^o)$ for the Almond case. Concerning the choice of a difficult right-hand side, (θ, ϕ) is fixed to $(90^o, 20^o)$ for the Aircraft case, $(65^o, 0^o)$ for the Cetaf case, $(25^o, 0^o)$ for the Cobra case and $(90^o, 40^o)$ for the Almond case. The symbol "-" means that convergence is not obtained within 5000 iterations. As can be seen, the smaller the restart, the larger the improvement with $M_{SLRU(k)}$. With $M_{SLRU(20)}$ on the Cetaf example, we

see that GMRES converges quickly and that GMRES(10) behaves very similarly to GMRES(∞). This observation is no longer true for the other geometries. It might depend on the number of eigenvalues that are computed: the choice is the best for the Cetaf example but not for the other geometries. The $M_{SLRU(k)}$ preconditioner allows us to use a smaller restart than usual; this might

Easy Case								
m	Cetaf		Aircraft		Cobra		Almond	
	M_{Frob}	$M_{SLRU(20)}$	M_{Frob}	$M_{SLRU(20)}$	M_{Frob}	$M_{SLRU(15)}$	M_{Frob}	$M_{SLRU(60)}$
10	271	30	-	639	514	378	492	95
30	128	27	-	439	280	196	149	67
50	100	27	-	390	264	188	118	68
∞	74	27	421	242	222	172	109	63
Difficult Case								
m	Cetaf		Aircraft		Cobra		Almond	
	M_{Frob}	$M_{SLRU(20)}$	M_{Frob}	$M_{SLRU(20)}$	M_{Frob}	$M_{SLRU(15)}$	M_{Frob}	$M_{SLRU(60)}$
10	669	37	-	1200	2624	481	-	1497
30	275	31	-	688	1031	232	429	163
50	197	31	-	608	760	207	334	144
∞	98	31	490	283	367	187	232	126

Table 4.9: Sensitivity of the restart parameter m on the number of iterations of the restarted GMRES using M_{Frob} and $M_{ISLRU(k)}$ on the four geometries, both on an easy and a difficult right-hand side. The tests were run on 8 processors of the Compaq machine.

be a significant asset on very large problems where using a large restart becomes a severe bottleneck because of the prohibitive memory requirements. There are some problems where M_{Frob} does not converge, for instance on the Aircraft case with a restart between 10 and 50, or on the Almond example with a restart of 10, while $M_{SLRU(k)}$ does converge in a reasonable number of iterations. On that latter example, as we see in Figure 4.6, the convergence rate of GMRES increases with the restart whatever the chosen preconditioner. Furthermore, the slope of the convergence history for $M_{SLRU(k)}$ becomes quickly comparable to that of full GMRES, while for M_{Frob} this phenomenon takes more time to appear. Although not reported here, the same behaviour was observed on the other geometries.

4.3.6 Results with the Flexible-GMRES method

In this section, we do not consider the multiple right-hand sides case as we intend just to illustrate a possible effect of the spectral low rank update on the numerical behaviour of the Flexible-GMRES method. The embedded scheme FGMRES(m)/GMRES(p) described in Section 3.4 is used, where m is the size of the outer restart and p the number of the full inner iterations allowed. An accurate matrix-vector product is used for each outer iteration and a fast matrix-vector product for each inner iteration. We then need eigen information from the matrix AM_{Frob} where A is defined by the fast matrix-vector product. This eigen information is again computed using ARPACK.

We keep the same easy and difficult angles per geometry as those defined in Section 4.3.5. Table 4.10 displays the number of accurate matrix-vector products of the outer system and the corresponding elapsed time to solve an easy and a difficult angle, using M_{Frob} or M_{SLRU} to precondition the inner system. In the column "# Mat.V","$x+y$" indicates x outer (i.e. accurate) matrix-vector products and y is the cumulated number of inner (i.e. fast) matrix-vector products. We keep the same number of eigenvectors per geometry for applying M_{SLRU} as in the previous section. That is, 20 for the Aircraft and the Almond case, and 15 for the Cobra case. With M_{SLRU} , the number of accurate matrix-vector products is reduced by a factor between 1.9 and

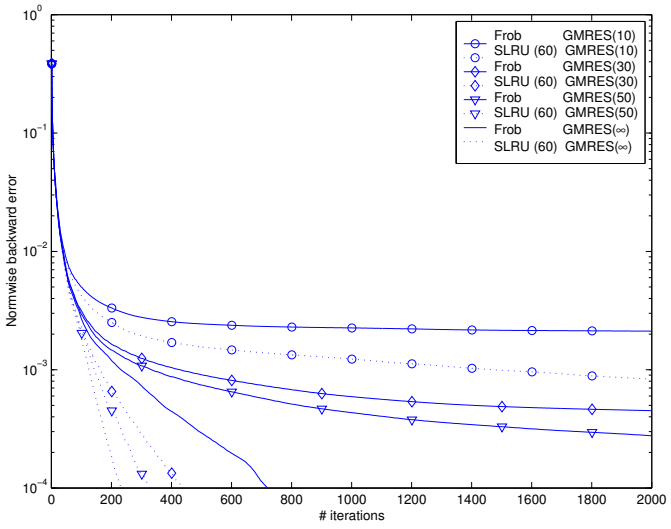


Figure 4.6: Convergence history varying the restart for a "difficult" angle of the Almond test problem.

2.4 on the difficult angle, and a factor between 1.2 and 2 for the easy angle. For the elapsed time we observe the same trend but not exactly with the same ratio as we use two accuracies for the matrix-vector product. The difference between easy and difficult angles tends to disappear with the use of M_{SLRU} . This observation has already been noticed in Figure 4.3 on the Cobra example once enough eigenvalues have already been shifted.

Aircraft							
Easy Case				Difficult Case			
FGMRES(30)/GMRES(60)				FGMRES(30)/GMRES(60)			
M_{Frob}		$M_{SLRU(20)}$		M_{Frob}		$M_{SLRU(20)}$	
# Mat.V	Time	# Mat.V	Time	# Mat.V	Time	# Mat.V	Time
24+1380	40 mn	12+660	19 mn	29+1680	45 mn	12+660	19 mn
Almond							
Easy Case				Difficult Case			
FGMRES(10)/GMRES(20)				FGMRES(10)/GMRES(20)			
M_{Frob}		$M_{SLRU(20)}$		M_{Frob}		$M_{SLRU(20)}$	
# Mat.V	Time	# Mat.V	Time	# Mat.V	Time	# Mat.V	Time
8+140	9 mn	6+100	6 mn	17+300	19 mn	9+160	10 mn
Cobra							
Easy Case				Difficult Case			
FGMRES(10)/GMRES(20)				FGMRES(10)/GMRES(20)			
M_{Frob}		$M_{SLRU(20)}$		M_{Frob}		$M_{SLRU(20)}$	
# Mat.V	Time	# Mat.V	Time	# Mat.V	Time	# Mat.V	Time
16+280	11 mn	14+240	9 mn	32+580	22 mn	14+240	9 mn

Table 4.10: Number of accurate matrix-vector products and elapsed time required to converge, using the restarted Flexible-GMRES method with M_{Frob} and $M_{SLRU(k)}$ for the inner GMRES solver, on an easy and a difficult right-hand side. The tests were run on 8 processors of the Compaq machine.

4.3.7 Complementarity of $M_{SLRU(k)}$ and the Seed-GMRES algorithm

As mentioned already for a complete monostatic calculation, several linear systems with the same coefficient matrix but different right-hand sides have to be solved. In that framework, it is crucial not only to use an efficient preconditioner but also a suitable Krylov solver. There are basically two classes of techniques designed for this purpose. These are the seed techniques and the block approaches [36, 67, 88]. In this section, we investigate the benefit of using $M_{SLRU(k)}$ in the context of seed GMRES.

The seed GMRES algorithm is described in Section 2.4. In order to illustrate the advantage of using $M_{SLRU(k)}$ in a multiple right-hand sides context, we consider eleven right–hand sides: $\phi = 15^o : 1^o : 25^o$ for $\theta = 90^o$, involved in the radar cross section calculation of the Aircraft test problem. On these right–hand sides, seed GMRES combined with M_{Frob} behaves rather poorly. To illustrate this, we first compare the numerical behaviour of GMRES with three strategies for defining the initial guess: first using the zero vector, second using the solution of the previous linear system, and finally the initial guess computed by the seed technique. In Table 4.11, we display the number of iterations for each right–hand side for the three initial guess strategies. In the case of zero for the initial guess, the initial backward error $\|r_0\|_2/\|b\|_2 = \|b - Ax_0\|_2/\|b\|_2$ is equal to one.

M_{Frob} preconditioner					
	zero guess	simple strat.		seed GMRES	
(θ, φ)	# iter	# iter	$\|r_0\|_2/\|b\|_2$	# iter	$\|r_0\|_2/\|b\|_2$
(90,15)	474	474	1	474	1
(90,16)	474	443	0.284	440	0.13
(90,17)	483	435	0.298	338	0.05
(90,18)	491	442	0.311	387	0.023
(90,19)	491	438	0.325	458	0.008
(90,20)	490	438	0.338	543	0.004
(90,21)	492	436	0.352	605	0.003
(90,22)	497	418	0.365	599	0.003
(90,23)	499	424	0.379	589	0.003
(90,24)	499	431	0.392	601	0.003
(90,25)	499	406	0.406	573	0.003
# iterations	5389	4785		5607	
elapsed time (s)	4 h 50 mn	4 h 17 mn		8 h	

Table 4.11: Number of iterations per right–hand side using three strategies for the initial guess on the Aircraft example with the M_{Frob} preconditioner on 8 processors.

In Table 4.11, we see that the seed GMRES method does a good job of decreasing the initial residual norm; it is always by far the smallest. Unfortunately, starting from the seed initial guess that is the closest (in the backward error sense) to the solution does not guarantee fast convergence. That is, from that good initial guess, GMRES performs rather poorly. It performs only slightly better than starting from zero and is outperformed by the approach that starts from the initial guess provided by the simple strategy of using the solution to the previous system. In other words and quite surprisingly, the approach that gives the smallest initial residual norm is not the method that gives the smallest number of iterations.

We have observed this behaviour of the seed GMRES method on some other difficult problems. Intuitively, it seems to us that an analogy exists between this behaviour and the observed stagnation of the classical restarted GMRES method. In the two cases, an initial guess is extracted from a Krylov space to generate a new Krylov space. As illustrated in the previous section, one possible remedy for the restarted GMRES method is to replace M_{Frob} by $M_{SLRU(k)}$.

In Table 4.12, we investigate this possibility. We keep the same strategies as in Table 4.11 but now use $M_{SLRU(20)}$ rather than M_{Frob}. The initial guess computed by the seed GMRES method becomes the strategy with the best performance. In Figure 4.7, for each right–hand side, $\phi = 15^o : 1^o : 25^o$, we plot the convergence history of the seed GMRES method with the two preconditioners. It can be seen that although the norm of the initial residuals are about the same for the two preconditioners, the rate of convergence is significantly improved by $M_{SLRU(20)}$. The seed GMRES method provides a small initial residual and $M_{SLRU(k)}$ ensures a fast rate of convergence of GMRES iterations: in a race starting close to the finish (seed strategy) and running fast ($M_{SLRU(k)}$ preconditioner) this ensures we finish first!

$M_{SLRU(20)}$ preconditioner					
	zero guess	simple strat.		seed GMRES	
(θ, φ)	# iter	# iter	$\|r_0\|_2/\|b\|_2$	# iter	$\|r_0\|_2/\|b\|_2$
(90,15)	280	280	1	280	1
(90,16)	275	198	0.284	201	0.14
(90,17)	275	188	0.298	145	0.052
(90,18)	276	190	0.311	167	0.025
(90,19)	280	198	0.325	165	0.009
(90,20)	283	205	0.338	171	0.006
(90,21)	284	208	0.352	171	0.005
(90,22)	286	212	0.365	170	0.005
(90,23)	289	214	0.379	172	0.005
(90,24)	291	218	0.392	176	0.005
(90,25)	292	219	0.406	171	0.005
# iterations	3111	2330		1989	
elapsed time (s)	2 h 19 mn	1 h 44 mn		1 h 32 mn	

Table 4.12: Number of iterations per right–hand side using three strategies for the initial guess on the Aircraft example with the $M_{SLRU(20)}$ preconditioner on 8 processors.

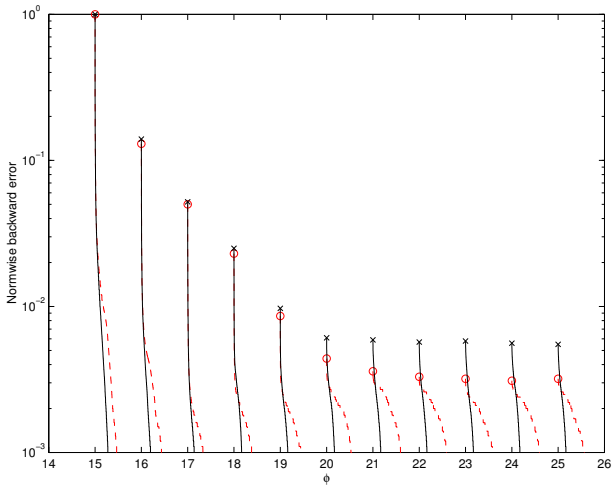


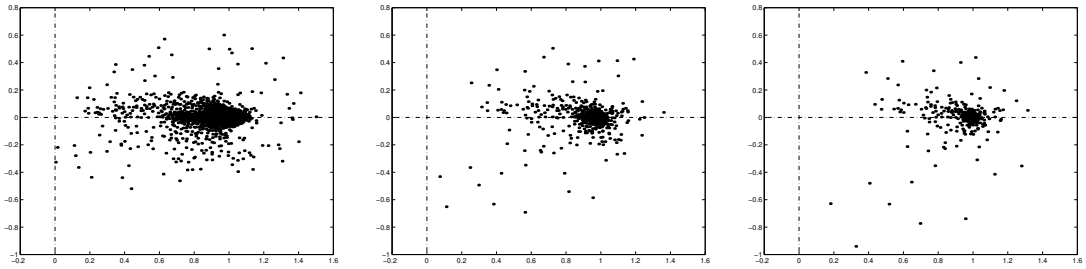
Figure 4.7: Convergence history of seed GMRES for the right-hand sides associated with $\phi = 15^o : 1^o : 25^o$ using M_{Frob} ($\bigcirc$, in dashed line) and $M_{SLRU(20)}$ ($\times$, in solid line) on the Aircraft test problem.

4.3.8 Efficiency of the additive spectral two-level preconditioner

In this section, we investigate the numerical efficiency of the M_{Add} preconditioner. We use a fast FMM calculation in the preconditioning operation during the smoothing iterations. The preconditioner M_{Add} requires eigenvectors of the matrix $M_{Frob}A$. We keep the same ARPACK basis formed by the right eigenvectors of AM_{Frob} with an accurate matrix-vector product, but we multiply it by M_{Frob} to obtain right eigenvectors of $M_{Frob}A$. For these numerical experiments, we setup ω such that:

$$\omega = \frac{3}{2}|\lambda_{max}(M_{Frob}A)|^{-1}.$$

This seems to be a good choice overall. The value $|\lambda_{max}|$ is estimated by ARPACK during the calculation of the eigenvectors in the preprocessing phase. We construct M_{Add} from a left preconditioner but we apply it as a right preconditioner. In Figure 4.8, we plot the spectrum of AM_{Add} on the Cetaf problem of size 5 391, varying the number μ of smoothing iterations for a low frequency correction based on twenty eigenvectors of $M_{Frob}A$. For the same value of μ, the preconditioner M_{Add} applies the same transformation as the preconditioner M_{Iter} on all eigenvalues except the twenty closest to zero. These latest eigenvalues are shifted to one thanks to the low frequency correction implemented by M_{Add}.



(a) $\mu = 1$ (b) $\mu = 2$ (c) $\mu = 3$

Figure 4.8: Spectrum of AM_{Add} for different values of μ on the Cetaf 5391. All the low frequency corrections are based on twenty eigenvectors of $M_{Frob}A$.

In Table 4.13 we report on the number of accurate matrix-vector products and the elapsed time obtained with full and restarted GMRES using the M_{Frob} and the M_{Add} preconditioners on the Cobra case of size 60 695 and on the Almond case of size 104 793. We consider a difficult angle for both cases: $(90^o, 140^o)$ for the Almond case and $(25^o, 0^o)$ for the Cetaf case. For the use of the preconditioner M_{Add}, we vary the size of the spectral correction and the number μ of smoothing iterations. It can be seen that the use of M_{Add} is always beneficial both from a number of iterations point of view but also from an execution time viewpoint. Using GMRES(10) on the Cobra problem, we observe a reduction by a factor of 12 between the elapsed time obtained by M_{Frob} and the elapsed time obtained by M_{Add} with 3 smoothing iterations and 15 eigenvectors. On the Almond problem, GMRES(10) preconditioned with M_{Frob} does not converge, whereas a combination with M_{Add} requires 6 mn to achieve convergence using 3 smoothing iterations and 50 eigenvectors.

From a numerical point of view, we observe on these examples that, the larger the size of the correction, the better the preconditioner; and that the number of matrix-vector products decreases when the number μ of smoothing iterations increases. Furthermore, the gain is larger if restarted GMRES is considered than if full GMRES is used as the solver. With $\mu = 3$ and 15 eigenvectors for the Cobra case or 50 eigenvectors for the Almond case, GMRES(10) and full GMRES tend to have the same performance in terms of number of matrix-vector products and elapsed time. We mention that on large electromagnetic problems (of size larger than 0.5 million unknowns) the use of small

restarts is recommended in order to save the heavy cost of reorthogonalization and to reduce the final solution cost. The choice of a small restart is also dictated by memory constraints [20]. With full GMRES, the number of matrix-vector products is significantly decreased but the total solution cost is likely to only slightly decrease when the number of smoothing iterations is large because the preconditioner is expensive to apply. The optimal selection of the size of the spectral correction and the number of smoothing iterations remains an open question; the choice mainly depends on the clustering properties of the initial preconditioner.

In terms of computational cost, the two transformations are complementary components that have to be suitably combined. On the Cobra problem, whatever the chosen GMRES, using $\mu = 3$ and 5 eigenvectors we obtain a better convergence rate but a similar computational time to using $\mu = 1$ with 15 eigenvectors. Enabling a reduction in the number of eigenvectors by using more smoothing iterations is a desirable feature in the context where computing many eigenvalues can become very expensive.

4.4 Conclusions

In this chapter, we have shown that the $M_{SLRU(k)}$ preconditioner based on the M_{Frob} preconditioner is effective for solving large linear systems arising in challenging real-life electromagnetics applications. It is important to point out that, to be effective, the spectral low rank update should be built on top of a good enough preconditioner that already succeeds in clustering most of the eigenvalues close to a point far from the origin (one in our case). There are two main reasons that motivate this observation. Firstly, if only a few eigenvalues are left close to the origin, a small rank update will be sufficient to significantly improve the convergence. Secondly, these few isolated eigenvalues will be easily found by an eigensolver. Of course a trade-off between the two components of $M_{SLRU(k)}$ should be found as the low rank update might be unnecessary if M_{Frob} is very dense, or it might be too expensive to improve a poor M_{Frob} because too many eigenvalues, potentially difficult to compute, have to be shifted.

We observe that the convergence of GMRES using $M_{SLRU(k)}$ only weakly depends on the choice of the initial guess. This is particularly useful in the seed GMRES or restarted GMRES contexts. Moreover, the benefit of using $M_{SLRU(k)}$ increases as the size of the restart decreases. In an embedded scheme, using a better preconditioner for the inner iterations is also beneficial both in elapsed time and number of outer iterations.

The M_{Add} preconditioner also turns out to be very effective. On the one hand, the eigenvalues close to zero are shifted to one. On the other hand, the smoothing operation acts on the rest of the spectrum and succeeds in clustering its main part also close to one. These two transformations are complementary. For difficult problems where computing many eigenvalues is expensive, this approach seems to be quite interesting as it can require less eigenvectors.

When several right-hand sides have to be solved with the same coefficient matrix, the extra cost of the eigencalculation in a preprocessing phase is quickly amortized by the reduction in the number of iterations as the extra cost per iteration is negligible. However, the cost of the preprocessing phase can be decreased in different ways. A first approach, as the accuracy of the eigenvectors in terms of backward error does not need to be very good, would be to still compute them in a preprocessing phase but with a less accurate FMM. Implementing this idea for the Cobra example leads to the plot reported in Figure 4.9. In that figure, it can be seen that using a low accuracy FMM only deteriorates the efficiency of the $M_{SLRU(k)}$ preconditioner by less than 20% in terms of number of iterations. Consequently, using a fast FMM for the eigencalculation might also be a way of reducing the cost for the preprocessing phase. Another possibility would consist in constructing the preconditioner as we solve different right-hand sides by extracting the spectral information from previous GMRES solutions. A question that remains open is the a priori identification of the optimal number of eigenvalues to be shifted. We have seen that increasing this number is always beneficial, but the relative gain tends to vanish when this number becomes large.

Cobra problem of size 60 695			
With M_{Frob}	GMRES(10) 2871 (59 mn)		
	GMRES(∞) 368 (14 mn)		
$\mu = 1$			
	Dimension of the coarse space		
	5	10	15
GMRES(10)	1492 (31 mn)	556 (12 mn)	510 (11 mn)
GMRES(∞)	262 (9 mn)	216 (7 mn)	188 (6 mn)
$\mu = 2$			
	Dimension of the coarse space		
	5	10	15
GMRES(10)	471 (15 mn)	209 (7 mn)	201 (7 mn)
GMRES(∞)	161 (7 mn)	132 (6 mn)	115 (5 mn)
$\mu = 3$			
	Dimension of the coarse space		
	5	10	15
GMRES(10)	281 (12 mn)	132 (6 mn)	124 (5 mn)
GMRES(∞)	120 (6 mn)	98 (5 mn)	85 (5 mn)

Almond problem of size 104 793			
With M_{Frob}	GMRES(10) *+3000*		
	GMRES(∞) 242 (14 mn)		
$\mu = 1$			
	Dimension of the coarse space		
	10	30	50
GMRES(10)	*+3000*	*+3000*	1867 (1 h)
GMRES(∞)	229 (13 mn)	157 (9 mn)	132 (8 mn)
$\mu = 2$			
	Dimension of the coarse space		
	10	30	50
GMRES(10)	494 (25 mn)	231 (12 mn)	168 (9 mn)
GMRES(∞)	133 (10 mn)	91 (7 mn)	76 (6 mn)
$\mu = 3$			
	Dimension of the coarse space		
	10	30	50
GMRES(10)	205 (14 mn)	110 (8 mn)	85 (6 mn)
GMRES(∞)	95 (9 mn)	65 (7 mn)	55 (6 mn)

Table 4.13: Number of accurate matrix-vector products and elapsed time required to converge on a single right-hand for the Cobra and Almond examples using GMRES(10) and full-GMRES solvers. Different values for the number μ of smoothing iterations and for the size of the coarse space are investigated. The tests were run on 16 processors of the Compaq machine.

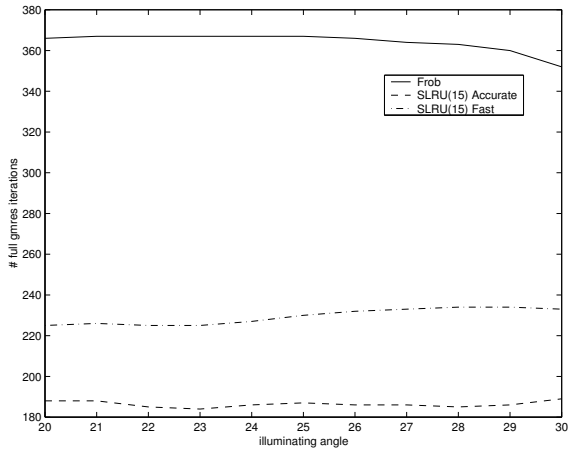


Figure 4.9: Influence of the multipole precision used in the calculation of the eigenvectors on the iterations on a difficult range for the Cobra test problem of size 60 695.

III

Chapter 5

Preconditioners based on a series of low-rank updates

5.1 Introduction

In this chapter, we study a solution technique suited for the solution of a set of large linear systems involving the same coefficient matrix but different right-hand sides. That is, the solution of

$$Ax^{(\ell)} = b^{(\ell)} \text{ for } \ell = 1, \ldots, \tag{5.1}$$

where A is a nonsingular matrix in $\mathbb{C}^{n\times n}$, and $x^{(\ell)}$ and $b^{(\ell)}$ are vectors of $\mathbb{C}^n$. Several numerical techniques can be considered to attempt to reduce the cost of solving subsequent systems in the sequence. The approach to follow might consist in changing the Krylov solver or in improving the preconditioner.

Sophisticated changes to the Krylov solver can be envisaged ranging from the seed approach, as described in Section 2.4 for GMRES where the initial guess vector is chosen so that it complies with an optimum norm or an orthogonality criterion over the Krylov space associated with the previous right-hand sides, to the more elaborate approaches such as GCRO-DR recently proposed in [68] that further exploits deflating ideas present in GMRES-E [63] or GMRES-DR [64]. The underlying idea in these latter techniques is to recycle Krylov vectors to build the space where the minimal residual norm solution will be searched for the subsequent system as described in Section 2.2. The GCRO-DR method is a flexible variant of the GMRES-DR method capable of Krylov subspace recycling between linear systems. GMRES-DR and GMRES-E cannot be modified to do so [68].

Other possible complementary alternatives consist in improving a selected preconditioner. In most of the situations, the linear systems are solved using an application dependent preconditioner whose efficiency and cost are controlled by a few parameters. Because the preconditioner is used for all the right-hand sides, some extra effort can be dedicated to improve it. The extra work involved in its construction can be easily amortized as many systems have to be solved. For instance, if an incomplete factorization [12, 45, 75] is considered, the threshold parameter can be decreased to allow for more fill-in in the factors giving rise to a more efficient preconditioner but more costly to build and to apply. This latter behaviour has been observed for instance in Table 4.8 of Section 4.3.4 in the electromagnetism calculation using M_{Frob}. Similarly, in the algebraic multigrid context we might decide to select techniques that lead to better preconditioners but that have a more expensive setup phase.

Even though such an approach is certainly beneficial, the gain is often limited and other complementary techniques can be envisaged. In particular, after each linear solution we might attempt to extract from the Krylov subspace approximations to some of the eigenvectors to update the preconditioner for the subsequent right-hand sides. We investigate a combination of a low rank

update spectral preconditioner and a Krylov solver that computes on the fly approximations of the eigenvectors associated with the smallest eigenvalues. We consider in this chapter the unsymmetric case but most of the techniques we describe in the sequel have a counterpart in the Hermitian positive definite situation.

In Section 5.2, we describe how the updates can be taken into account when extra eigen information is extracted after each linear solution. From one linear system to the next, a low rank correction is added to the preconditioner and we therefore use the terminology "incremental preconditioner". We discuss in Section 5.3 the choice of a linear Krylov solver that provides us with the spectral information we are seeking. We show that all the ingredients involved both in the definition and in the implementation of the incremental spectral preconditioner are by-products of the selected Krylov solver. In Section 5.4 we illustrate on academic problems some of the typical observed behaviours for the proposed numerical scheme. Section 5.5 is devoted to the implementation of our solution technique in the large parallel industrial code presented in Section 1.3.3. For that real life application, significant gains in computational time on parallel computers are obtained thanks to the incremental preconditioner approach.

5.2 Incremental spectral preconditioners

5.2.1 Formulations and properties

In this section, we provide a heuristic motivation for our incremental preconditioning methods. In Chapter 4, a spectral low rank update (SLRU) preconditioning technique that exploits the knowledge of the eigenvectors associated with the smallest eigenvalues has been described. The proposed preconditioners shift the corresponding eigenvalues close to one and numerical examples show the relevance of this approach to speed-up the convergence of Krylov solvers. We consider now a repeated use of the spectral preconditioner.

Let assume that we solve the first linear system using a left preconditioned Krylov solver and we denote by M the initial preconditioner. Let $U^{(1)}$ be the basis of a right invariant subspace of MA of dimension k_1, such that $MAU^{(1)} = U^{(1)}J_k$, where the eigenvalues of J_k are $\{\lambda_1, \cdots, \lambda_{k_1}\}$. From Proposition 4.2.1, we can define a preconditioner $M^{(2)}_{ISLRU}$ so that $\{\lambda_1, \cdots, \lambda_{k_1}\}$ will be shifted in the spectrum of the matrix $M^{(2)}_{ISLRU}A$, involved in the solution of the second linear system:

$$M^{(2)}_{ISLRU}Ax^{(2)} = M^{(2)}_{ISLRU}b^{(2)}.$$

Similarly after the solution of this system, we suppose that we obtain $U^{(2)}$ associated with the eigenvalues $\{\lambda_{k_1+1}, \cdots, \lambda_{k_2}\}$ so that these eigenvalues will be shifted in the spectrum of the matrix $M^{(3)}_{ISLRU}A$, involved in the solution of the third linear system:

$$M^{(3)}_{ISLRU}Ax^{(3)} = M^{(3)}_{ISLRU}b^{(3)}.$$

Notice, under the assumptions of Proposition 4.2.1, we have the following expression for that latter preconditioner:

$$M^{(3)}_{ISLRU} = M^{(2)}_{ISLRU} + U^{(2)}\left(W^{(2)^H}AU^{(2)}\right)^{-1}W^{(2)^H}.$$

We repeat this procedure until step ℓ and assume that once the ℓ^{-th} linear system has been solved, we not only have the solution $x^{(\ell)}$ but we also know the $(k_\ell - k_{\ell-1})$ vectors of the set $U^{(\ell)}$ that span the invariant subspace associated with the $(k_\ell - k_{\ell-1})$ eigenvalues $\{\lambda_{k_{\ell-1}+1}, \cdots, \lambda_{k_\ell}\}$ of $M^{(\ell)}_{ISLRU}A$. We can then update the preconditioner $M^{(\ell)}_{ISLRU}$ so that the eigenvalues of $M^{(\ell+1)}_{ISLRU}A$ are:

$$\begin{cases} \eta_i^{(\ell+1)} = 1 + \lambda_i & \text{if} \quad i \leq k_\ell, \\ \eta_i^{(\ell+1)} = \lambda_i & \text{if} \quad i > k_\ell. \end{cases} \tag{5.2}$$

The preconditioner for the $(\ell+1)^{-th}$ linear system can be written as:

$$M_{ISLRU}^{(\ell+1)} = M + \sum_{j=1}^{\ell} U^{(j)} ({W^{(j)}}^H A U^{(j)})^{-1} {W^{(j)}}^H. \tag{5.3}$$

Assume that a basis $V^{(1)}$ of a right invariant subspace of AM (and no longer MA) associated with $\{\lambda_1, \cdots, \lambda_{k_1}\}$ is given. Let us consider now a right preconditioned Krylov solver to deal with problem (5.1). Using a similar incremental process as before, under the assumptions of Proposition 4.2.2, the preconditioner for the $(\ell+1)^{-th}$ linear system is:

$$M_{ISLRU}^{(\ell+1)} = M + \sum_{j=1}^{\ell} M_{ISLRU}^{(j)} V^{(j)} ({W^{(j)}}^H A M_{ISLRU}^{(j)} V^{(j)})^{-1} {W^{(j)}}^H. \tag{5.4}$$

Using Proposition 4.2.2, it can also be shown by induction that the eigenvalues of $AM_{ISLRU}^{(\ell+1)}$ are given by (5.2). For any ℓ, if $V^{(\ell)}$ is a basis of a right invariant subspace of $AM_{ISLRU}^{(\ell)}$ associated with $\{\lambda_{k_{l-1}+1}, \cdots, \lambda_{k_l}\}$, $M_{ISLRU}^{(\ell)} V^{(\ell)}$ is a basis of the right invariant subspace of $M_{ISLRU}^{(\ell)} A$ associated with the same eigenvalues. By setting $U^{(\ell)} = M_{ISLRU}^{(\ell)} V^{(\ell)}$ for any ℓ, Equation (5.3) becomes equivalent to Equation (5.4). As mentioned earlier the slow convergence of Krylov solvers is often attributed to the presence of small eigenvalues in the spectrum of the preconditioned matrix. In that context, it is natural to target for $V^{(\ell)}$ a right invariant subspace associated with the eigenvalues close to the origin.

As we plan to compare the numerical efficiency obtained by an initial preconditioner M with the numerical efficiency obtained by the incremental preconditioner based on M, we consider a right preconditioning version. The sketch of the solution scheme for a sequence of right-hand sides in MATLAB like syntax is described in Algorithm 7 for the case where the basis of the right invariant subspaces associated with any $AM_{ISLRU}^{(\ell)}$ are available. Algorithm 7 is by no means well suited for practical implementation. The computation of an invariant subspace is generally far more expensive than the solution of a linear system and the low rank-update as performed by step 5 would generally fill the preconditioner which is unacceptable. The purpose of the next section is to show how this algorithm can be adapted so that it is suitable to practical implementation.

Algorithm 7 Basic scheme for a sequence of right-hand sides.

1: $M_{ISLRU}^{(1)} = M$
2: **for** $\ell = 1, 2, \ldots$ **do**
3: $[x^{(\ell)}]$ =Solve($b^{(\ell)}$, $M_{ISLRU}^{(\ell)}$, A)
4: $[V^{(\ell)}]$ =Right_Invariant_Space($AM_{ISLRU}^{(\ell)}$)
5: $M_{ISLRU}^{(\ell+1)} = M_{ISLRU}^{(\ell)} \left(I + V^{(\ell)} \left({W^{(\ell)}}^H A M_{ISLRU}^{(\ell)} V^{(\ell)} \right)^{-1} {W^{(\ell)}}^H \right)$
6: **end for**

5.3 A Krylov linear solver recovering spectral information

The spectral preconditioning techniques are likely to be particularly efficient if they are implemented to complement a prescribed ad-hoc and efficient preconditioner that only leaves a few outliers near the origin. Because our primary interest is to solve a sequence of linear systems we would like to recover the eigen information almost for free; that is, either because it is a by-product of the linear solver or because it can be computed at a low computational cost from information already computed by the linear solver. In that context, natural candidates among the Krylov linear solvers are those that rely on an Arnoldi procedure and belong to the variants of GMRES. In

particular, because we are looking for the smallest eigenvalues and because for large scale computation a restart mechanism has to be implemented, GMRES-DR(m, k) [64] appears as an appealing candidate.

This solver provides us with a nice trade-off between its efficiency to solve a linear system and its ability to compute approximations of the eigenvectors associated with the smallest eigenvalues as a by-product of the method. This latter feature is due to the harmonic restarted Arnoldi technique that it implements. At each restart of size m, GMRES-DR (m, k) computes the k smallest harmonic Ritz vectors of the $(m+1) \times m$ Hessenberg matrix that are used to span a subspace of the Krylov space to be explored in the next restart. We refer the reader to Section 2.2 for more details, and to [64] for a complete description.

Because GMRES-DR exhibits nice capabilities to recover harmonic spectral information we are targeting with our preconditioner, we explore in the rest of the study the numerical behaviour of the combination of our incremental technique with this Krylov linear solver. In practice, the columns of $V^{(\ell)}$ are the right eigenvectors associated with the k smallest eigenvalues and we set $W^{(\ell)} = V^{(\ell)}$ for our experiments. The resulting implementation of the numerical method is obtained by replacing steps 3 and 4 of Algorithm 7 by a call to GMRES-DR where of course the preconditioner M_{ISLRU} is kept in implicit form. That is, it is never assembled and whenever a preconditioning operation is required we only have to perform matrix-vector products involving $V^{(\ell)}$ and to solve small linear systems involving matrices ${V^{(\ell)}}^H A M^{(\ell)}_{ISLRU} V^{(\ell)}$. These matrices can be cheaply recovered from the GMRES-DR process without any extra matrix-vector product by A or $M^{(\ell)}_{ISLRU}$ as is shown in the next section.

Because we stop the GMRES-DR iterations when the scaled residual $\eta_b(x_k)$ of the linear systems is less than a prescribed tolerance ε, there is no guarantee that the spectral information has converged at the same time. The quality of an approximate eigenvector is checked via the backward error defined by Equation (2.12). This eigenvector backward error is estimated via the upper bound defined in Equation (2.14). This upper bound is cheap to compute because it uses by-products of the GMRES-DR algorithm.

5.4 Analysis of the incremental mechanism

5.4.1 Some computational considerations

We see in Algorithm 7, that after each right-hand side the preconditioner is updated. Even though the preconditioner is kept in its implicit form, the update requires the calculation and the LU factorization of a $k_\ell \times k_\ell$ matrix $A_c^{(\ell)} = {V^{(\ell)}}^H A M^{(\ell)}_{ISLRU} V^{(\ell)}$, where k_ℓ denotes the rank of the update to be performed after the solution of the ℓ^{-th} right-hand side. Similarly to the analysis we made in Section 4.2.3 we give below some details on the computational cost in terms of floating-point operations and memory requirements associated with the incremental preconditioner. We assume that $k_\ell \ll m$ so that any terms that are only polynomial in k_ℓ are neglected; we only keep the terms in m with the highest degree. Because the approximations of the eigenvectors are recovered by GMRES-DR, the matrix $A_c^{(\ell)}$ can be cheaply computed. Using the same notation as in Section 2.2 and denoting by $G^{(\ell)}$ the $m \times k_\ell$ matrix whose columns are the g_i used to construct the harmonic Ritz eigenvectors (see Section 2.2.1), we have:

$$\begin{aligned} A_c^{(\ell)} &= {V^{(\ell)}}^H A M^{(\ell)}_{ISLRU} V^{(\ell)} \\ &= {G^{(\ell)}}^H {Q_m^{(\ell)}}^H A M^{(\ell)}_{ISLRU} Q_m^{(\ell)} G^{(\ell)} \\ &= {G^{(\ell)}}^H H_m^{(\ell)} G^{(\ell)}. \end{aligned} \tag{5.5}$$

Using Equation (5.5) the calculation of $A_c^{(\ell)}$ costs $\mathcal{O}(m^2 k^2)$ operations and its LU factorization that requires $\frac{2}{3} {k_\ell}^3$ flops is negligible.

The application of the preconditioner reduces to the application of a sequence of M_{SLRU} preconditioners whose individual cost is given in Table 4.1. It reads:

$$2nnz(M) + 4n\sum_{j=1}^{\ell} k_j.$$

From a memory point of view, we have to store $\sum_{j=1}^{\ell} k_j$ vectors of dimension n plus ℓ small matrices of size $k_j \times k_j$.

5.4.2 Selection of the initial guess

When a sequence of linear systems has to be solved where the right-hand sides do not vary a lot from one to the next, it can be natural to setup the initial guess for a given right-hand side as a function of the solution of the previous linear system. Let $x^{(\ell-1)}$ be the solution of the $(\ell-1)^{-th}$ linear system and $x_0^{(\ell)}$ the initial guess of the ℓ^{-th} system. The implementation of this idea is straightforward when a left preconditioner is used as it is enough to set $x_0^{(\ell)} = x^{(\ell-1)}$. For a right preconditioner, this approach requires us to pay a little more attention and might not be feasible. In this case, we solve $AMt = b$ and we have access to the matrix-vector product with M. For the initial guess we have to choose $t_0^{(\ell)}$ so that $Mt_0^{(\ell)} = x^{(\ell-1)}$ or equivalently $t_0^{(\ell)} = M^{-1}x^{(\ell-1)}$ that requires the ability to perform a matrix vector product by M^{-1} or to solve the linear system associated with M. Consequently for a right preconditioner, the implementation of this strategy for computing the initial guess requires us to have access to both M and M^{-1}. This is the case of the implicit preconditioners based on incomplete factorizations like ILU [75], $AINV$ [11, 12], $FSAI$ [51, 52], ... In these cases, it is as simple to perform a backward/forward substitution as to perform two triangular matrix-vector products. For our incremental right preconditioner, this idea can also be implemented under the assumption that both M and M^{-1} are available. This reduces to defining $t_0^{(\ell)}$ so that it is a solution of:

$$x_0^{(\ell)} = M_{ISLRU}^{(\ell)} t_0^{(\ell)} = x^{(\ell-1)}.$$

Because M_{ISLRU}, defined by Equation (5.4), is a series of low-rank updates, it is possible to get an expression for M_{ISLRU}^{-1} using a sequence of Sherman-Morrison formulae [40]. The resulting expression is given in the next proposition.

Proposition 5.4.1 *Assume that the initial preconditioner M as well as all the matrices $A_c^{(i)} + V^{(i)^H}V^{(i)}$, with $i \in \{1, ..., \ell-1\}$ for $\ell \geq 2$ are nonsingular. Then $M_{ISLRU}^{(\ell)}$ is nonsingular and its inverse is:*

$$\left(M_{ISLRU}^{(\ell)}\right)^{-1} = \prod_{i=1}^{\ell-1} \left(I - V^{(\ell-i)}\left(A_c^{(\ell-i)} + V^{(\ell-i)^H}V^{(\ell-i)}\right)^{-1} V^{(\ell-i)^H}\right) M^{-1}$$

Proof:
This statement can be proved by induction. Let $D \in \mathbb{C}^{n\times k}$, $E \in \mathbb{C}^{k\times k}$ and $F \in \mathbb{C}^{n\times k}$. Assume that $(E^{-1} + FD)$ are nonsingular, the Sherman-Morrison formula gives a convenient expression for the inverse of $(I + DEF)$:

$$(I + DEF)^{-1} = I - D(E^{-1} + FD)^{-1}F\,.$$

If we take $\ell = 2$, $D = V^{(1)}$, $E = \left(A_c^{(1)}\right)^{-1}$ and $F = V^{(1)^H}$: $(E^{-1} + FD) = A_c^{(1)} + V^{(1)^H}V^{(1)}$. This quantity is nonsingular by assumption, then:

$$\left(I + V^{(1)}\left(A_c^{(1)}\right)^{-1}V^{(1)^H}\right)^{-1} = I - V^{(1)}\left(A_c^{(1)} + V^{(1)^H}V^{(1)}\right)^{-1}V^{(1)^H}, \tag{5.6}$$

and $M^{(2)}_{ISLRU}$ is nonsingular. In this way, we can write:

$$\begin{aligned} t_0^{(2)} &= \left(M^{(2)}_{ISLRU}\right)^{-1} x^{(1)} \\ &= \left(I + V^{(1)}\left(A_c^{(1)}\right)^{-1} {V^{(1)}}^H\right)^{-1} M^{-1}x^{(1)} \\ &= \left(I - V^{(1)}\left(A_c^{(1)} + {V^{(1)}}^H V^{(1)}\right)^{-1} {V^{(1)}}^H\right) M^{-1}x^{(1)}. \end{aligned}$$

Induction on ℓ enables us to complete the proof.

□

The above result indicates that the calculation of $t_0^{(\ell)}$ requires a product by M^{-1} and $(\ell-1)$ constructions and factorizations of small matrices: $A_c^{(\ell-i)} + {V^{(\ell-i)}}^H V^{(\ell-i)}$. We omit the details and only give the overall floating-point cost associated with the calculation of $t_0^{(\ell)}$, that is:

$$2nnz(M) - n + 2n\sum_{i=1}^{\ell-1} k_i(k_i+2).$$

5.4.3 Sensitivity to the quality of the eigenvectors

The numerical motivations given in Section 5.2 for using the incremental scheme assume that the basis of the invariant space or the corresponding eigenvectors are known exactly. This does not make sense in finite-precision and is certainly too strong a constraint in our incremental preconditioning framework. However, a natural question is how the performance of the incremental spectral preconditioner is affected by approximate eigen information ? In order to investigate this, we compute the eigenpairs (using a backward stable eigensolver that is the `eig` function of MATLAB) of a slightly perturbed matrix, $(AM_1 + E)$, with $\frac{||E||}{||AM_1||} = \eta$, and we use these eigenvectors to build our preconditioners. By varying η, we can monitor the level of the backward error associated with each eigenvector.

We report on numerical experiments for two real unsymmetric matrices from Matrix Market namely HOR131 and GRE1107 with an initial preconditioner that is $ILU(t)$. We set $t = 2.10^{-1}$ for HOR131 and $t = 6.10^{-2}$ for GRE1107. The stopping criterion is based on $\eta_b(\tilde{x})$ defined by Equation (2.16) with a threshold equal to 10^{-8}. The right-hand side is obtained by multiplying the test matrix by a vector whose entries are equal to one. We measure the benefit of using the spectral preconditioner by considering the ratio between the number of GMRES-DR iterations with the spectral update divided by the number of iterations with only the $ILU(t)$ preconditioner. In Figure 5.1, we plot this ratio as a function of η for $M_{SLRU(4)}$ and $M_{SLRU(8)}$ used as a right preconditioner. Several comments are in order. Firstly we notice that the more eigenvalues that are shifted, the better the preconditioner. Secondly, it is clear that knowing the eigenvectors with full accuracy is not necessary. For instance a backward error of about 10^{-4} gives rise to a preconditioner that is as efficient as the one obtained with a backward error of about 10^{-16} for the HOR131 example. The range of backward errors that leads to preconditioners with optimum efficiency is problem dependent. We see the deteriorations of the ratio appear for different values of η for the two examples considered in the Figure 5.1. Furthermore, it can be seen that in terms of number of iterations using four "accurate enough" eigenvectors is as efficient as using eight less accurate ones. For instance on the HOR131 example, we have a 20 % gain either using four eigenvectors computed with a backward error of 10^{-3} or using eight eigenvectors computed with a backward error of 10^{-1}. Of course in terms of computational cost, it is more efficient to only use four eigenvectors as the application of the preconditioner will be cheaper. Finally, we see that very inaccurate eigen information does not help and even slightly deteriorates the convergence of the linear solver.

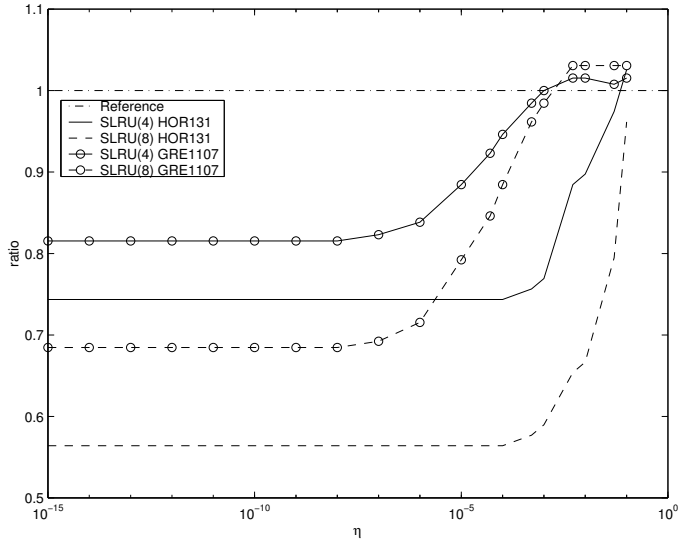


Figure 5.1: Sensitivity of spectral preconditioner varying the accuracy and the number of eigenvectors.

In an incremental mechanism, where the eigenvectors are extracted from the Krylov space built by the linear solver it might naturally happen that some of the eigenvectors are quite inaccurate. Using these vectors would increase the application cost of the preconditioner, without necessarily reducing the number of iterations. Consequently, we implement a criterion for the selection of the candidate vectors to be used for building the low rank updates. Only the approximate eigenvectors that have a backward error smaller than τ_ξ are selected. This criterion is based on the backward error associated with a normalized approximate eigenvector and is estimated using Equation (2.14). Furthermore, because we assume that the smallest eigenvalues are those that slow down the convergence the most, we only consider the candidate vectors that are associated with eigenvalues which have a magnitude smaller than a prescribed threshold τ_λ.

5.4.4 Results on a sequence of sparse linear systems

We report on the numerical experiments that have been performed in MATLAB using right preconditioners. We keep the same stopping criterion: it is based on $\eta_b(\tilde{x})$ with a threshold equal to 10^{-8}. We denote by M_{ILU} the initial preconditioner constructed using regular $ILU(t)$. We intend to compare the numerical behaviour of the incremental preconditioner M_{ISLRU} with M_{ILU}. In Table 5.1, we display the list of our test problems from Matrix Market; all these matrices are real nonsymmetric. We set the threshold t for M_{ILU} to illustrate the behaviour generally observed when the spectrum of the preconditioned system has only a few eigenvalues close to the origin. That is, when the initial preconditioner M_{ILU} is already effective in clustering most of the eigenvalues near one. For information, we also present the number of nonzero entries of the factors L and U resulting from the incomplete factorization of the matrix A, the number of nonzero entries of A as well as the domain of each matrix. We use GMRES-DR (m, k) with a restart $m = 30$ and a number of recycled eigenvectors $k = 5$. Because we setup our sequences of right-hand sides so that they do not vary much from one to the next we performed the experiments using the strategy described in Section 5.4.2 to define the initial guess. The selection of the candidate eigenvectors is based on the two thresholds $\tau_\lambda = 0.5$ and $\tau_\xi = 10^{-2}$. The sequence of right-hand sides is defined as:

$$b^{(i)} = b^{(i-1)} * (1 + \alpha * rand(n, 1)), \tag{5.7}$$

Matrix	Size	t	$nnz(L_A)$	$nnz(U_A)$	nnz(A)	Discipline
BWM200	200	6.10^{-1}	200	200	796	Chemical engineering
BFW398A	398	5.10^{-1}	412	416	3678	Electrical engineering
HOR131	434	2.10^{-1}	819	1468	4182	Flow in networks
PORES3	532	5.10^{-1}	705	874	3474	Reservoir modeling
ORSIRR1	1 030	3.10^{-1}	1648	1838	6858	Oil reservoir simulation
GRE1107	1 107	2.10^{-2}	33435	31587	5664	Computer systems simulation
BWM2000	2 000	4.10^{-1}	3992	4106	7996	Chemical engineering
RDB2048	2 048	1.10^{-2}	36941	36857	12032	Chemical engineering

Table 5.1: Set of real nonsymmetric test matrices.

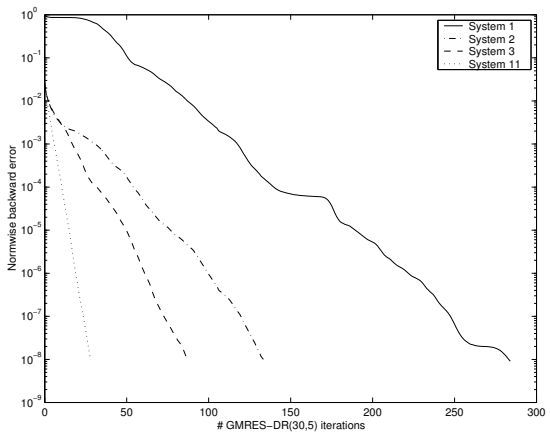
where $b^{(i)}$ is the i^{-th} right-hand side, and α a real parameter controlling the gap between the right-hand sides. For the experiments reported in this manuscript we consider two values for α: 10^{-1} and 10^{-4}.

To illustrate the numerical benefit introduced by the incremental preconditioner, we first plot in Figure 5.2 (resp. in Figure 5.3) the backward error histories obtained with the matrix BWM200 for a sequence of eleven right-hand sides setup with $\alpha = 10^{-1}$ (resp. with $\alpha = 10^{-4}$). More precisely, we display in Figure 5.2 (a) and Figure 5.3 (a) the convergence history observed using M_{ISLRU}, and in Figure 5.2 (b) and Figure 5.3 (b), the convergence history observed using M_{ILU}. For the M_{ISLRU} preconditioner, each increment is at most a rank-5 update of the preconditioner, that corresponds to the value of k used for GMRES-DR. The benefit of using the incremental spectral preconditioner is clear because performing the update always implies a reduction in the number of GMRES-DR iterations. It translates into a better convergence rate for GMRES-DR. In Figure 5.2 (a) the fastest convergence is obtained when 32 eigenvalues are shifted for the 11^{-th} right-hand side, whereas in Figure 5.2 (b) it is for a total of 28 eigenvalues.

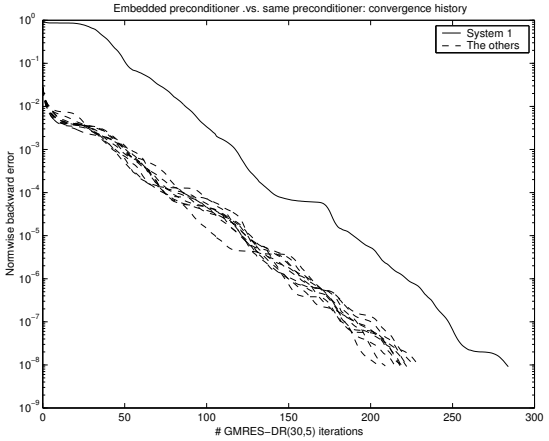
In these figures, we can also observe the effect of the initial guess strategy. For the M_{ILU} preconditioner, the strategy consists in using the solution of the previous system to define the initial guess for the subsequent system. For the M_{ISLRU} preconditioner, the initial guess is deduced from the solution of the previous system using Proposition 5.4.1. We see that whatever the preconditioner is, the initial backward error for the first right-hand sides is one (because we choose $x_0 = 0$ for the first right-hand side) while this initial backward error is much smaller for the subsequent right-hand sides. This reflects the fact that the first residuals are much smaller. We also observed that this gap is larger when the right-hand sides, and consequently the solutions, are closer than those generated with $\alpha = 10^{-1}$. We also see in Figure 5.2 (b) and Figure 5.3 (b) that, for M_{ILU}, the rate of convergence remains the same for all the right-hand sides, while it improves with M_{ISLRU} as illustrated in Figure 5.2 (a). To illustrate the overall gain introduced by M_{ISLRU} versus M_{ILU} on this sequence of right-hand sides, for $\alpha = 10^{-1}$, we need 2470 iterations to solve all the systems using M_{ILU}, and 837 iterations using $M_{ISLRU(32)}$: it saves 66% of the iterations. With $\alpha = 10^{-4}$ we need 1505 iterations to solve all the systems using M_{ILU}, and 637 iterations using $M_{ISLRU(28)}$: it saves 58% of the iterations.

In Figure 5.4 (a) (resp. Figure 5.5 (a)), we depict the number of GMRES-DR iterations when the right-hand sides are generated using $\alpha = 10^{-1}$ (resp. $\alpha = 10^{-4}$) for the matrix BFW398A. The solid line gives the number of iterations observed using M_{ILU} as a preconditioner and the dashed line gives the number of iterations when using M_{ISLRU}. The gap between the two curves reveals the significant gain in number of iterations introduced by M_{ISLRU}. In Figure 5.4 (b) and Figure 5.5 (b) we indicate how the number of shifted eigenvalues increases as the right-hand sides are solved. The steps indicate that, after each solution, not all of the five candidate eigenvectors comply with the magnitude (τ_λ) or backward error (τ_ξ) constraints.

However, from a computational point of view the picture is slightly different as illustrated

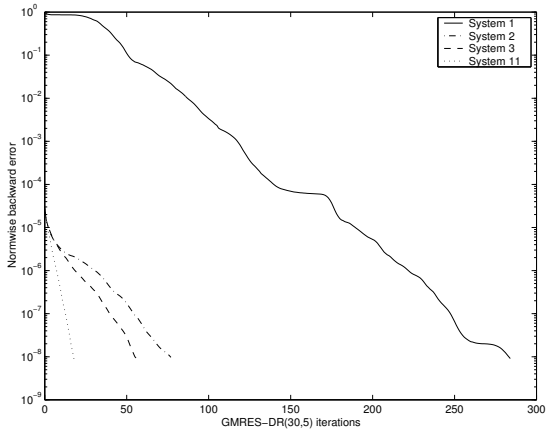


(a) backward errors using M_{ISLRU}

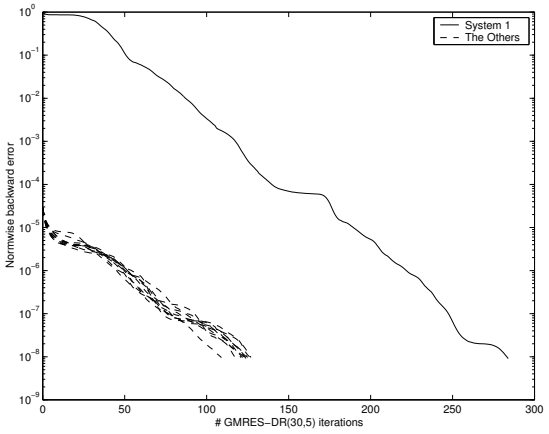


(b) backward errors using M_{ILU}

Figure 5.2: Convergence histories of some systems using M_{ISLRU} (a) and M_{ILU} (b) with $\alpha = 10^{-1}$ on the matrix BWM200.



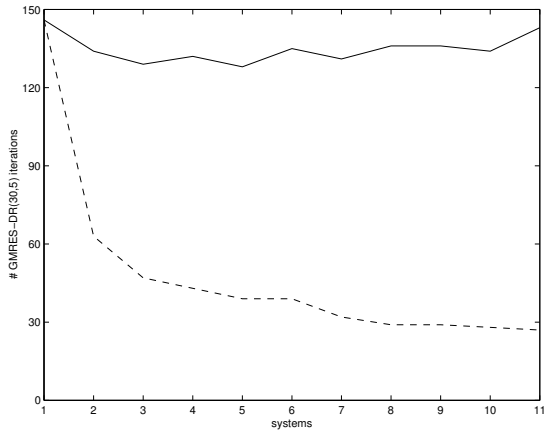
(a) backward errors using M_{ISLRU}



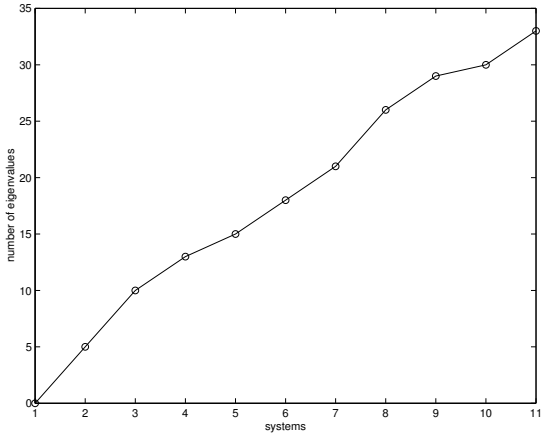
(b) backward errors using M_{ILU}

Figure 5.3: Convergence histories of some systems using M_{ISLRU} (a) and M_{ILU} (b) with $\alpha = 10^{-4}$ on the matrix BWM200.

in Figure 5.6 (a) and Figure 5.7 (a). Always for the matrix BFW398A, these figures display for each system the number of millions of floating-point operations ('MFlops') required to achieve convergence using M_{ILU} (solid line) and M_{ISLRU} (dashed line) for a sequence of 51 right-hand sides generated either with $\alpha = 10^{-1}$ or $\alpha = 10^{-4}$. Because we keep incrementing the preconditioner, the cost per iteration increases as we solve the right-hand sides. Consequently, although we decrease the number of iterations, we do not reduce in the same proportion the number of floating-point operations and hence the elapsed time. At a certain point the MFlops per right-hand side starts to grow. The update of the preconditioner does not improve the convergence much (i.e. the number of iterations does not decrease) but the weight of each iteration increases. This can be observed in particular in Figure 5.6 (a) where, at the end of the sequence, 81 eigenvalues have been shifted. This behaviour also appears in Figure 5.7 (a) with $\alpha = 10^{-4}$ but is more moderate. In that latter case, only 43 eigenvalues have been shifted because the convergence of GMRES-DR, thanks to the better efficiency of the initial guess strategy, is faster so that less

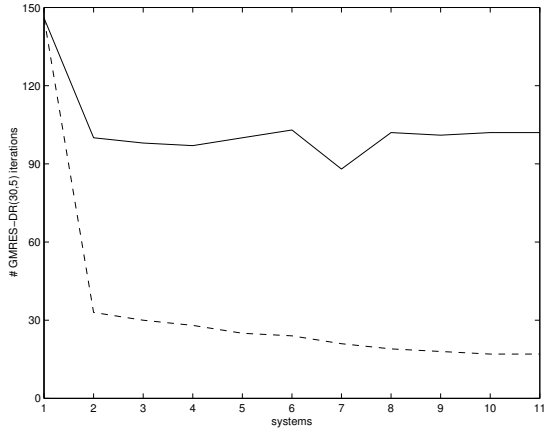


(a) Number of iterations with M_{ISLRU} and M_{ILU} for a sequence of 11 right-hand sides.

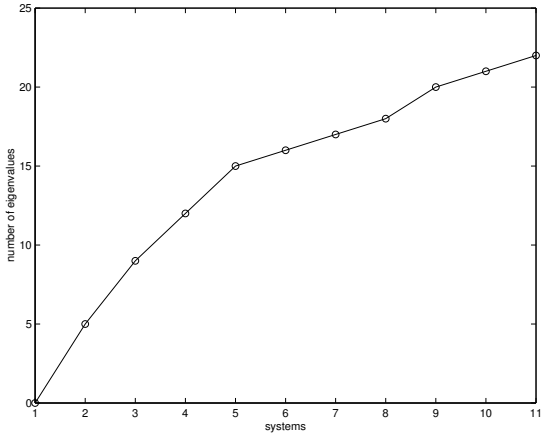


(b) Number of shifted eigenvalues.

Figure 5.4: Efficiency of $M_{ISLRU(k)}$ in decreasing the number of iterations (a) when varying k (b) as the right-hand side changes with $\alpha = 10^{-1}$ on the matrix BFW398A.



(a) Number of iterations with M_{ISLRU} and M_{ILU} for a sequence of 11 right-hand sides.

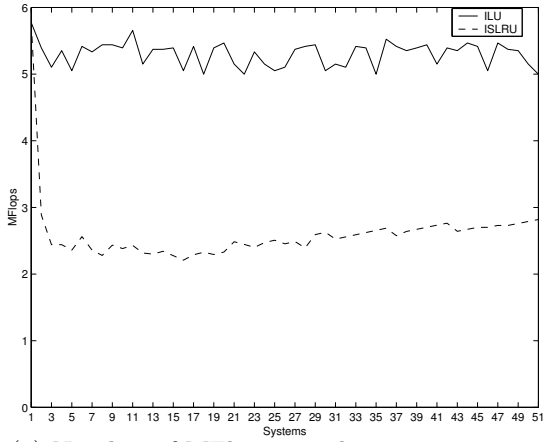


(b) Number of shifted eigenvalues.

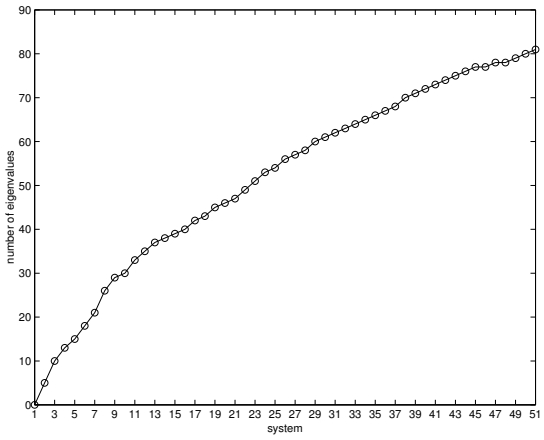
Figure 5.5: Efficiency of $M_{ISLRU(k)}$ in decreasing the number of iterations (a) when varying k (b) as the right-hand side changes with $\alpha = 10^{-4}$ on the matrix BFW398A.

eigenvectors comply with the magnitude and backward error criteria. Finally, we mention that, from a memory point of view, the preconditioner requires more space as increments are performed and it would become unaffordable at some point.

In Tables 5.2 and 5.3 we summarize and compare the performance of M_{ILU} and M_{ISLRU} for the two sets of right-hand sides using or not using the initial guess strategy described in Section 5.4.2. A sequence of 31 right-hand sides is considered. We see in these tables that M_{ISLRU} always outperforms M_{ILU} in terms of number of iterations. Even though the ratio is smaller with respect to the number of floating-point operations than with respect to the number of iterations, M_{ISLRU} also outperforms M_{ILU} for all examples but ORSIRR1. Both the fact that the floating-point ratio is smaller than the number of iterations ratio and the fact that M_{ILU} outperforms M_{ISLRU} on ORSIRR1 is due to the relative high cost of M_{ISLRU} compared to the other numerical

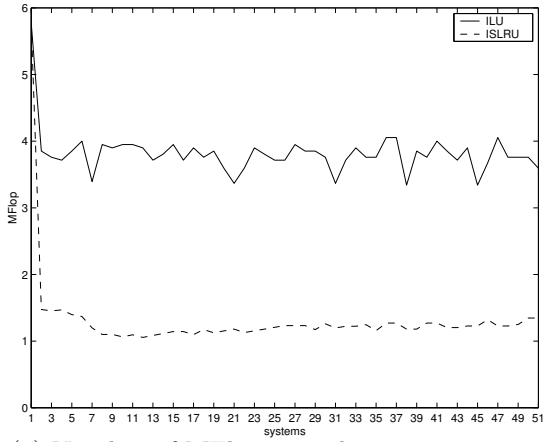


(a) Number of MFlops to achieve convergence for each linear solver using M_{ILU} and M_{ISLRU}.

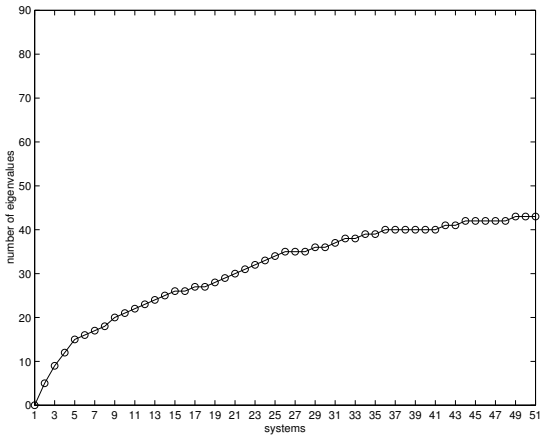


(b) Number of shifted eigenvalues.

Figure 5.6: Number of MFlops and number of shifted eigenvalues on a sequence of 51 right-hand sides with $\alpha = 10^{-1}$ on the matrix BFW398A.



(a) Number of MFlops to achieve convergence for each linear solver using M_{ILU} and M_{ISLRU}.



(b) Number of shifted eigenvalues.

Figure 5.7: Number of MFlops and number of shifted eigenvalues on a sequence of 51 right-hand sides with $\alpha = 10^{-4}$ on the matrix BFW398A.

kernels involved in GMRES-DR.

Let us illustrate this phenomenon on the ORSIRR1 case. As can be shown in Figure 5.8 (a), the spectrum of AM_{ILU}, with A the ORSIRR1 matrix, has many eigenvalues in the neighbourhood of zero (around 20 % of the total). The rest of the eigenvalues are clustered in the neighbourhood of one except one eigenvalue near 1.5. A large sequence of 90 right-hand sides is considered for this experiment. As depicted in Figure 5.8 (b), the incremental procedure performs a rank-5 update between each right-hand side, until approximately the 40^{-th} where the size of the forthcoming updates decreases until it vanishes at the 47^{-th} right-hand side. The impact on the number of iterations is displayed in Figure 5.8 (c), where we compare the number of iterations to achieve

convergence using either M_{ILU} or M_{ISLRU}. The preconditioner M_{ISLRU} succeeds in decreasing the number of iterations, but the gain in iterations from one system to the next remains constant, at about 10%. We have to wait for the 30^{-th} right-hand side to obtain a significant decrease in the number of iterations. Figure 5.8 (d) shows the number of MFlops required by M_{ILU} and M_{ISLRU} to achieve convergence. From the first 29^{-th} right-hand side, the reduction in the number of iterations is not sufficient to balance the higher cost of M_{ISLRU} that shifts 5 eigenvalues at each right-hand side. Consequently, the number of MFlops is still growing and M_{ILU} is the best approach. From the 30^{-th} system, as enough small eigenvalues have been shifted, a significant decrease in the number of iterations results in a more significant decrease in the MFlops. The trend for the MFlops is reversed and the M_{ISLRU} approach eventually outperforms the M_{ILU} preconditioner.

Matrix	Iterations			MFlops	$k_{\max}$	RHS
	M_{ILU}	M_{ISLRU}	Ratio	Ratio		α
BWM200	4 031	932	4.3	2.7	40	10^{-4}
	6 866	1 245	5.5	3	53	10^{-1}
BFW398A	3 090	672	4.6	2.9	37	10^{-4}
	4 112	941	4.4	2.1	62	10^{-1}
HOR131	2 012	626	3.2	1.3	63	10^{-4}
	3 264	895	3.6	1.5	62	10^{-1}
PORES3 *	8 244	1 289	6.4	4.2	63	10^{-4}
	10 380	1 615	6.4	3.9	60	10^{-1}
ORSIRR1	2 702	1 995	1.4	0.4	139	10^{-4}
	4 913	2 624	1.9	0.5	148	10^{-1}
GRE1107 *	4 316	1 017	4.2	3.4	33	10^{-4}
	5 418	1 206	4.5	3.6	30	10^{-1}
BWM2000	805	412	2	1.1	18	10^{-4}
	940	515	1.8	1.3	17	10^{-1}
RDB2048	2 188	811	2.7	2.1	22	10^{-4}
	3 880	1 010	3.8	3.2	19	10^{-1}

Table 5.2: Cumulated flops and iterations with M_{ILU} and M_{ISLRU} for all test problems using two sequences of 31 right-hand sides: $\alpha = 10^{-4}$ and $\alpha = 10^{-1}$. The initial guess strategy is used. The symbol ' * ' corresponds with the use of GMRES-DR(50,5) instead of GMRES-DR(30,5), in order to ensure convergence of the first system of the sequence.

Finally in Figure 5.9 and 5.10, we plot the spectrum of the preconditioned matrix $AM_{ISLRU(k)}$ through the sequence of right-hand sides for two test cases: BWM200 and PORES3. Because the eigenvectors are not computed exactly we see that the smallest eigenvalues (in magnitude) are not exactly shifted to one while the rest of the spectrum is unchanged. In Appendix A, for all the test cases described in Table 5.1, some spectrum of the preconditioned matrix $AM_{ISLRU(k)}$ through the sequence of right-hand sides are displayed. In most of the examples the complete spectrum is changed when spectral increments are applied to the preconditioner. However, these plots show that M_{ISLRU} does its job in "cleaning" the neighbourhood of the origin and in still clustering most of the eigenvalues close to one.

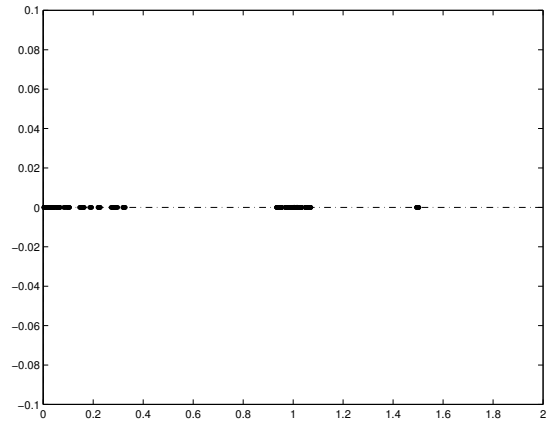
Matrix	Iterations			MFlops	$k_{\max}$	RHS
	M_{ILU}	M_{ISLRU}	Ratio	Ratio		α
BWM200	8 804	1 800	4.9	3	27	10^{-4}
	8 772	1 472	6	3.2	54	10^{-1}
BFW398A	4 526	1 083	4.2	2.2	41	10^{-4}
	4 572	1 054	4.3	2.1	59	10^{-1}
HOR131	3 875	1 022	3.8	1.8	58	10^{-4}
	3 899	1 045	3.7	1.8	55	10^{-1}
PORES3 *	16 886	1 765	9.6	6	60	10^{-4}
	16 708	1 731	9.7	6.3	57	10^{-1}
ORSIRR1	6 076	2 857	2.1	0.6	148	10^{-4}
	6 049	2 896	2.1	0.6	149	10^{-1}
GRE1107 *	5 580	1 170	4.8	4.4	18	10^{-4}
	5 542	1 204	4.6	4.1	29	10^{-1}
BWM2000	1 147	460	2.5	2.7	7	10^{-4}
	1 147	501	2.3	2.3	10	10^{-1}
RDB2048	5 301	1 458	3.6	3.1	10	10^{-4}
	5 031	1 458	3.5	3.1	10	10^{-1}

Table 5.3: Cumulated flops and iterations with M_{ILU} and M_{ISLRU} for all test problems using two sequences of 31 right-hand sides: $\alpha = 10^{-4}$ and $\alpha = 10^{-1}$. The initial guess is $x_0 = 0$. The symbol ' * ' corresponds with the use of GMRES-DR(50,5) instead of GMRES-DR(30,5), in order to ensure convergence of the first system of the sequence.

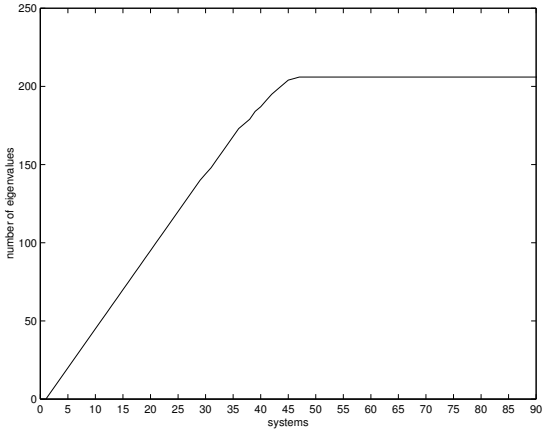
5.4.5 Controlling the number of vectors involved in the setup of M_{ISLRU}

The description we made so far of M_{ISLRU} assumes that we do not have any memory limit so that we can keep updating the preconditioner as long as candidate eigenvectors computed by GMRES-DR can be selected for one right-hand side to the next. This is clearly unrealistic and a mechanism must be defined to control the growth of M_{ISLRU}. We notice that this memory constraint is very similar to the one encountered with Krylov linear solvers like GMRES which has given rise to the restarted variants. Similarly to the solution that consists in limiting the size of the Krylov space, we limit the amount of storage we can afford for the incremental preconditioner. Since most of the memory for the incremental preconditioner is used for the storage of the vectors $V^{(\ell)}$, we control the memory required by the preconditioner by limiting the total number of these vectors. If k_{max} is the maximum number of vectors we can afford, we denote by $M_{ISLRU(k_{max})}$ the preconditioner that is eventually built once k_{max} vectors have been selected while solving the sequence of right-hand sides.

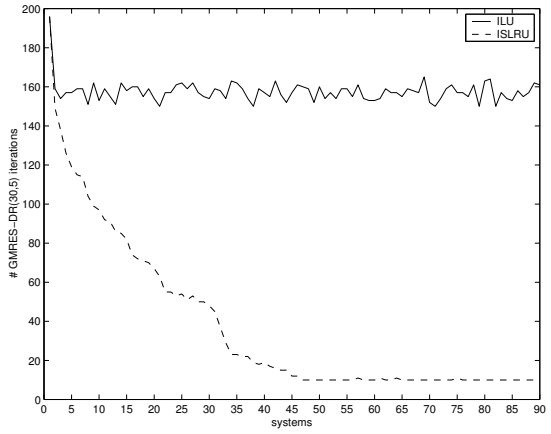
Different strategies can be considered to fill this memory space as we solve the sequence of right-hand sides. That is, we might want to take any candidate eigenvector that complies with the backward error and the eigenvalue magnitude criteria as long as there is still memory available or we might prefer to select only a subset of the candidate vectors. In particular, to have a better chance to capture the eigenvectors associated with the smallest eigenvalues, we might be tempted to select among the candidates only a subset of those associated with the smallest eigenvalues. In Figure 5.11, we illustrate the effect of various strategies to set-up a $M_{ISLRU(15)}$ preconditioner on the ORSIRR1 example, using a sequence of 31 right-hand sides defined by $\alpha = 100$ with the same initial guess strategy, and an initial preconditioner $ILU(3 \cdot 10^{-1})$. More precisely, among the up to five candidates computed by GMRES-DR($30, 5$) we either only select those associated with the smallest, or the three smallest or all the five eigenvalues.



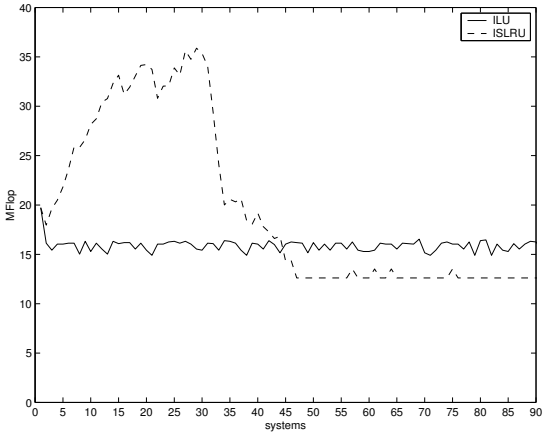
(a) Spectrum of AM_{ILU}



(b) Number of shifted eigenvalues per system



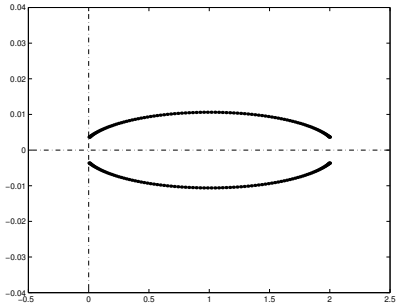
(c) Number of iterations to achieve convergence per system using M_{ILU} and M_{ISLRU}



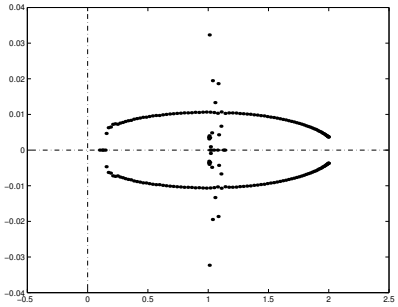
(d) Number of MFlops to achieve convergence per system using M_{ILU} and M_{ISLRU}

Figure 5.8: Efficiency of $M_{ISLRU(k)}$ in decreasing the number of iterations (c) and MFlops (d) when varying k (b) as the right-hand side changes, with $\alpha = 10^{-1}$ on the matrix ORSIRR1. The spectrum of AM_{ILU} is displayed in (a).

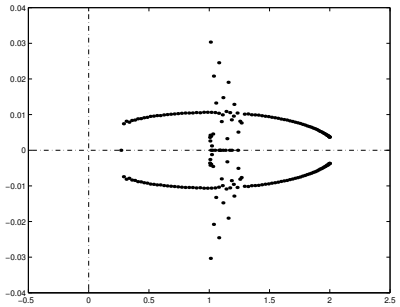
Two regimes can be identified in that figure. In the *start-up phase*, i.e. the first right-hand sides that enable us to build the $M_{ISLRU(15)}$ preconditioner, we can see that taking any values enables us to get the fastest convergence. For the first four right-hand sides, keeping five candidates per right-hand side is the most efficient. For the next nine, choosing three among the five becomes more efficient. In the *stationary phase*, i.e. k_{max} vectors have been collected by each of the three strategies, we observe that the strategy that took the longer to fill the memory allocated for the preconditioner is eventually the one that enables the fastest convergence. In that example, the strategy selecting one candidate per system turns out to be the best. In Figure 5.12, we give a heuristic explanation for that behaviour. In Figure 5.12 (a) we display, for the first three right-hand sides, the magnitude of the five approximate harmonic Ritz values computed by GMRES-DR (using the symbol " $\circ$ ") and the magnitude of the five smallest eigenvalues computed by `eig` (using the " $*$ " symbol) of the first three preconditioned matrices involved in the sequence. Similarly, in Figure 5.12 (b) we display the magnitude of the three harmonic Ritz values and exact eigenvalues for the first five preconditioned matrices. When looking at the magnitude of the 15 harmonic Ritz



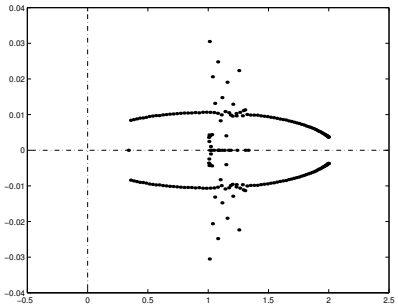
(a) 1^{-st} Syst – k = 0



(b) 10^{-th} Syst – k = 30

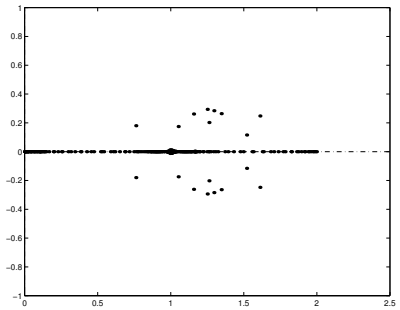


(c) 20^{-th} Syst – k = 47

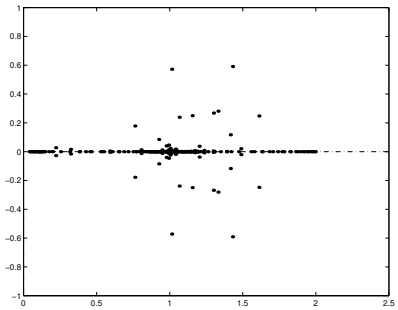


(d) 31^{-th} Syst – k = 53

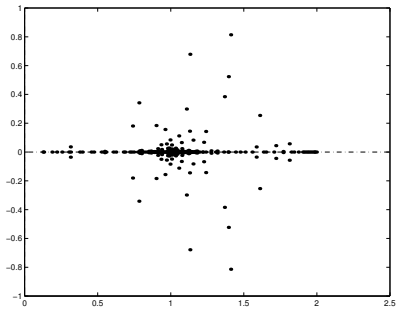
Figure 5.9: Spectrum of $AM_{ISLRU(k)}$ for different systems of the sequence, where A is the matrix BWM200.



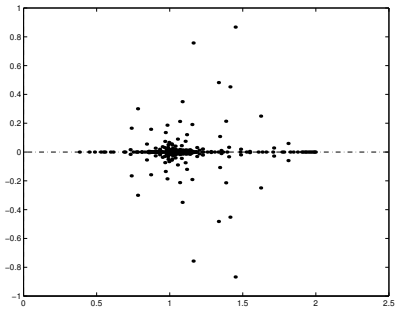
(a) 1^{-st} Syst – k = 0



(b) 5^{-th} Syst – k = 19



(c) 10^{-th} Syst – k = 44



(d) 15^{-th} Syst – k = 55

Figure 5.10: Spectrum of $AM_{ISLRU(k)}$ for different systems of the sequence, where A is the matrix PORES3.

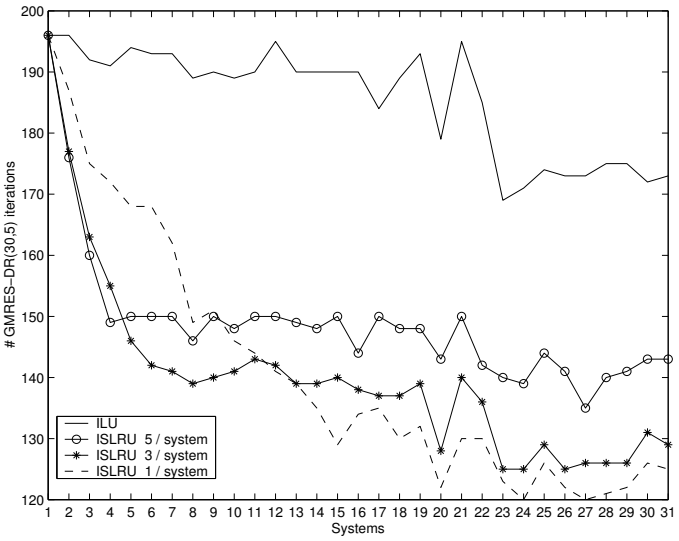


Figure 5.11: ORSIRR1, $\alpha = 100$: Number of iterations of GMRES-DR with three start-up strategies to build a $M_{ISLRU(15)}$ preconditioner.

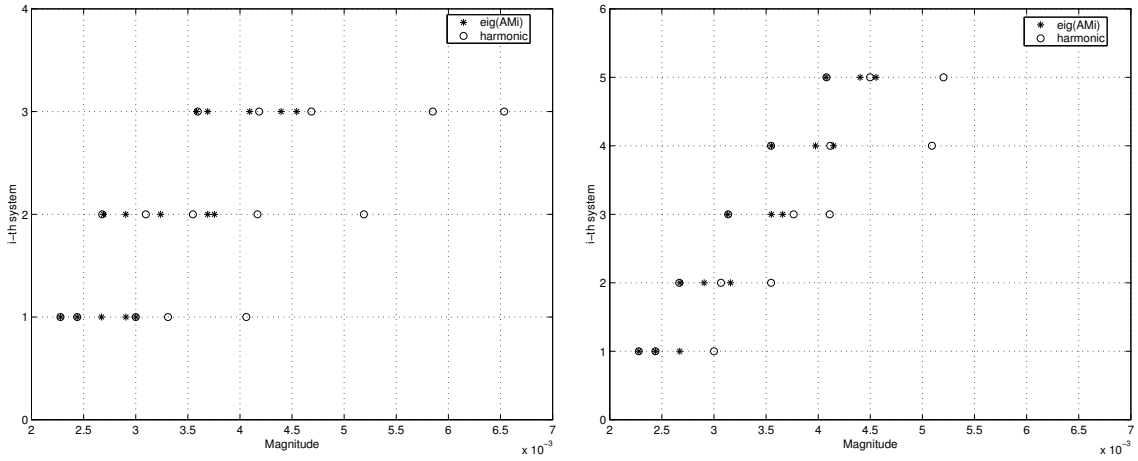
values shifted by each of the two strategies, we see that selecting only three eigenvectors per system enables us to shift smaller eigenvalues, which is the target of the spectral preconditioners. In all cases, the accuracy of the eigenvectors measured by their backward error are comparable for all the displayed harmonic Ritz values and are around 10^{-4} on the ORSIRR1 example. The final better performance exhibited by the incremental preconditioners built using a delaying strategy in the start-up phase has been observed on many of the examples we have considered. Consequently the selection strategy for a given sequence of right-hand sides should be a balance between the cost associated with the start-up phase (that is more expensive using the delaying strategy because the first linear systems are more costly to solve due to a "poorer" preconditioner) and the eventual gain in the stationary phase that is related to the overall number of right-hand sides. So if only a few right-hand sides have to be solved a short start-up phase (i.e. taking any candidate vector that complies with the magnitude and backward error conditions) should be advised; for large sequences of right-hand sides a delaying strategy should be preferred.

Finally, we performed experiments with various sets of right-hand sides, ranging from vectors slowly varying to orthogonal right-hand sides. The convergence histories were different but the general trends were the same; that is, the incremental preconditioner is always beneficial in terms of number of iterations, and implementing a delaying selection strategy to set-up the incremental preconditioner when the amount of memory is limited gives rise to more efficient preconditioners in the stationary phase.

5.5 Application to large electromagnetism calculations

5.5.1 Results on a monostatic calculation

For the experiments in electromagnetism we consider a radar cross section that is an angular section of width 30^o discretized every degree so that we end up with a sequence of 31 right-hand sides. The angular section of interest (θ, ϕ) is $(\, 90^o \,, 150^o - 180^0 \,)$ for the Aircraft geometry, $(\, 60^o - 90^o \,, 0^o \,)$ for the Cetaf geometry, and $(\, 60^o - 90^0 \,, 0^o \,)$ for the Cobra geometry. The initial guess strategy described in Section 5.4.2 can no longer be applied in this framework, because



(a) Magnitude of the approximate eigenvalues using 5 candidates per right-hand side

(b) Magnitude of the approximate eigenvalues using 3 candidates per right-hand side

Figure 5.12: ORSIRR1, $\alpha = 100$: Eigenvalues captured by the different strategies in the start-up phase.

the inverse of the preconditioner is required and the preconditioner M_{Frob} is only given in an implicit form. The initial guess is $x_0 = 0$ for all the experiments. The out-of-core capability significantly alleviates the memory constraint that previously motivated our choice to limit the amount of space to store the incremental preconditioner. We select any candidate eigenvector that has a backward error less than $\tau_\xi = 10^{-2}$ and a corresponding eigenvalue of magnitude smaller than $\tau_\lambda = 0.3$. In Figure 5.13, we display, for the large problems of each geometry presented in Section 1.3.2, the number of iterations with and without the incremental preconditioner on the sequence of right-hand sides. It can be seen that the incremental preconditioner enables us to significantly reduce the number of iterations. For the solution of the last right-hand side of the sequence, the reduction in GMRES-DR iterations is equal to about 10 for the Aircraft 94 000 example, about 4 for the Cetaf 264 000 and greater than 4 for the Cobra 179 000. In Table 5.4, we give more details on these numerical experiments. We report on the size of the problems for each geometry, the number of processors denoted by "# Proc", the size of the restart "m" and the number "k" of harmonic vectors required by the GMRES-DR solver, the cumulated number of matrix-vector products denoted by "# Mat.V" and the parallel elapsed time "Time" to perform the complete simulation. Finally, "$k_{\max}$" denotes the total number of eigenvalues shifted by the incremental preconditioner. We can see that the incremental mechanism enables us not only to reduce significantly the number of iterations but also the elapsed time.

The gain in time ranges from two to eight depending on the problem and is almost proportional to the reduction in the total number of iterations. This can be explained by the fact that the cost of a matrix-vector product is quite high and the relative cost of our preconditioner remains low even for large $k_{\max}$; consequently any reduction in the number of iterations translates to a reduction in the computational time. We notice that this property is not necessarily true for sparse linear systems, where the cost of the incremental preconditioner might dominate even for small values of $k_{\max}$ so that the preconditioner might not be effective if it does not significantly reduce the number of iterations.

In Table 5.5, we report on the parallel elapsed time required by each linear algebra kernel involved in the solution scheme. For M_{ISLRU} the application time corresponds to the time to apply the preconditioner for the last linear system; that is, when the preconditioner is the most expensive. We see that the application time, given in seconds, of the most expensive M_{ISLRU} is always significantly cheaper that the matrix-vector product implemented by the efficient parallel

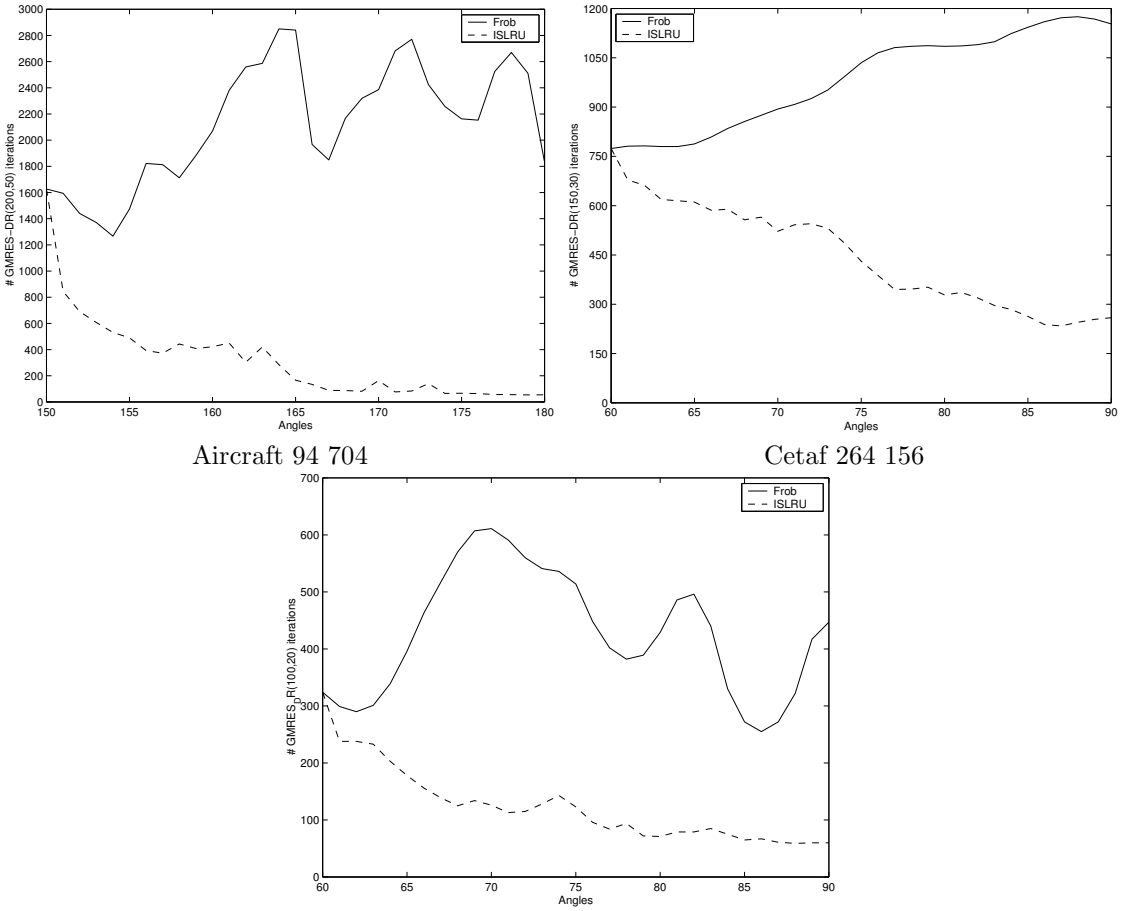


Figure 5.13: Number of GMRES-DR iterations with M_{Frob} and M_{ISLRU} for the different incident angles for each geometry. The sampling for the illuminating wave is one degree.

				M_{Frob}		M_{ISLRU}		
Geometry	Size	# Proc	(m,k)	# Mat.V	Time	# Mat.V	$k_{\max}$	Time
Aircraft	23 676	8	(200,50)	20 613	12h 15	2 688	246	1h 30
Aircraft	94 704	31	(200,50)	66 411	2d 21h	9 801	686	10h
Cetaf	86 256	31	(150,30)	23 047	14h 30	6 558	462	4h 30
Cetaf	134 775	31	(150,30)	22 254	23h 20	9 098	577	10h
Cetaf	264 159	31	(150,30)	30 804	2d 8h	13 921	770	1d 3h
Cobra	60 695	8	(100,20)	9 672	8h 30	2 092	200	2h
Cobra	179 460	31	(100,20)	13 418	14h	3 876	365	4h

Table 5.4: Cost for a monostatic calculation of an angular section of width 30^o.

FMM. This confirms that, if the preconditioner succeeds in decreasing the number of iterations, it will also decrease the total solution time.

In Figure 5.14, we give another view on the gain introduced by the incremental preconditioner

			Application Time (sec)		
Geometry	Size	# Proc	FMM	M_{Frob}	M_{ISLRU}
Aircraft	23 676	8	1.23	0.17	0.43
Aircraft	94 704	31	2.23	0.33	0.75
Cetaf	86 256	31	1.55	0.12	0.46
Cetaf	134 775	31	2.62	0.19	0.70
Cetaf	264 159	31	4.07	0.39	1.44
Cobra	60 695	8	2.16	0.21	0.66
Cobra	179 460	31	2.79	0.23	0.78

Table 5.5: Parallel elapsed time for each linear algebra kernel involved in the numerical scheme.

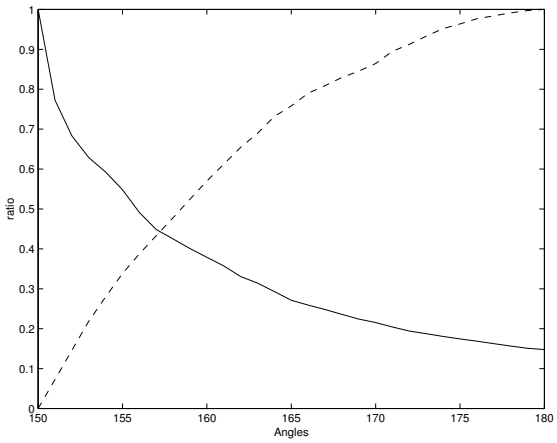
versus simple M_{Frob} in terms of elapsed time for the largest problems of each geometry. The solid line represents the ratio:

$$\frac{\sum_{i=1}^{\ell} T_i(M_{ISLRU}^{(i)})}{\sum_{i=1}^{\ell} T_i(M_{Frob})},$$

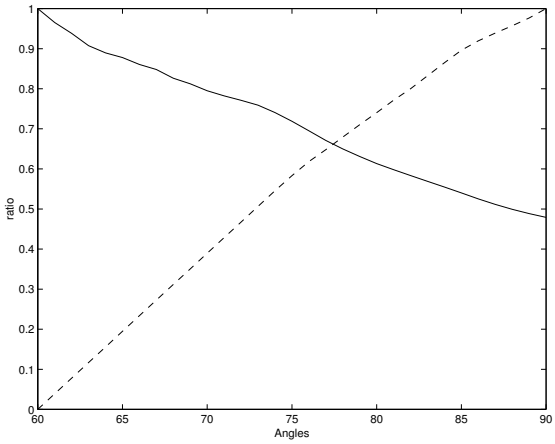
where $T_i(\mathcal{M})$ is the elapsed time to solve the i^{-th} system using the preconditioner $\mathcal{M}$. Furthermore, to give an idea about how the incremental preconditioner grows we give using a dashed line the percentage of the total shifted eigenvalues. This line illustrates that we have not limited the space allocated to the incremental preconditioner. We see that we keep updating the preconditioner until the last right-hand sides for the two largest examples. Those graphs illustrate the significant gains introduced by the incremental approach.

5.5.2 Limitation of the number of shifted eigenvalues

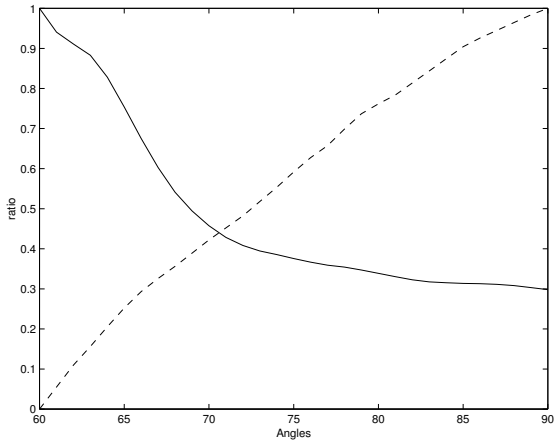
Even though in this out-of-core context the memory constraint is weaker, we report in Figure 5.15 on numerical experiments where we limit to 50 the number of approximate eigenvectors involved in the incremental preconditioner. We consider a radar cross section of width 90^o discretized every 3 degrees so that we end up with a sequence of 31 right-hand sides. We select the GMRES-DR(100,20) solver. As in the experiments of Section 5.4.5, we investigate the performance of the incremental preconditioner for different selection strategies. More precisely, we consider the strategies that take any eigenvectors that comply with τ_ξ and τ_λ criteria, and a strategy that only retains three eigenvectors associated with the smallest eigenvalues associated with the candidate eigenvectors. Similarly to what we observed in the previous section, it can be seen that delaying the fill-in of the incremental preconditioner enables us to end up with a more efficient preconditioner. We see in Figure 5.15 (a) that after the start-up phase, i.e. 4^{-th} system for the greedy approach and 17^{-th} system for the delaying strategy, this latter approach enables a larger reduction of the number of iterations. Furthermore, as illustrated in Figure 5.15 (b) the delaying strategy eventually outperforms the greedy strategy by about 10% in elapsed time. A heuristic explanation of this behaviour is that delaying the fill-in of the incremental preconditioner enables the possibility of capturing smaller eigenvalues. These eigenvalues are those that are suspected to contribute the most to slowing down the convergence of the Krylov solvers. This is illustrated for the Cobra example in Figure 5.16 where we display the magnitude of the 50 shifted eigenvalues using three strategies. It can be seen that the greedy approach tends to shift eigenvalues that are larger than those captured by the delaying strategies.



Aircraft 94 704



Cetaf 264 156



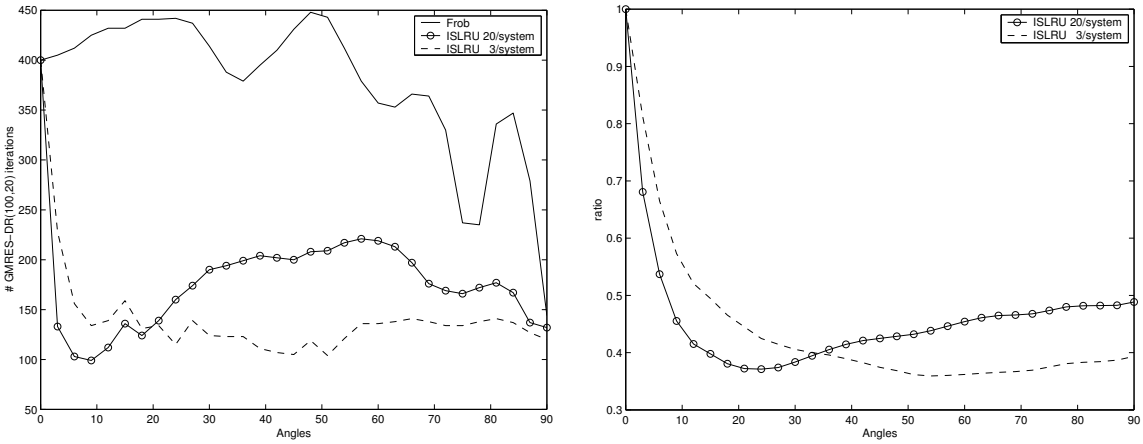
Cobra 179 460

Figure 5.14: Ratio of cumulated elapsed solution times (solid line) - percentage of shifted eigenvalues (dashed line).

5.6 Conclusions

In this last part, we have proposed a solution technique suited for the solution of a sequence of linear systems. This technique is a combination of a low rank update spectral preconditioner and a Krylov solver that recovers at run time approximations to the eigenvectors associated with the smallest eigenvalues, namely GMRES-DR in our case. As the eigenvectors are extracted from the Krylov space built by the linear solver, it is not surprising that we recover some inaccurate eigenvectors. Not only do these vectors increase the application cost and the memory requirement of the spectral preconditioner but they are likely to be inefficient at reducing the number of iterations. To prevent such a phenomenon, we consider a selection criterion based on the backward error of the eigenvector. As we target the eigenvalues of smallest magnitude, we define a selection criterion based on the magnitude.

We illustrate on a set of MATLAB examples the behaviour of this technique on academic sparse linear systems using an initial guess based on the previous solution. The gain in iterations is always larger than the gain in MFlops, due to the extra cost involved in the application of M_{ISLRU} , that



(a) On the impact of different start-up phases on the iterations

(b) On the impact of different start-up phases on the cumulated solution time

Figure 5.15: Number of iterations for two start-up strategies to build a $M_{ISLRU(50)}$ preconditioner on the COBRA 60695 example. The tests were run on 8 processors of the Compaq machine.

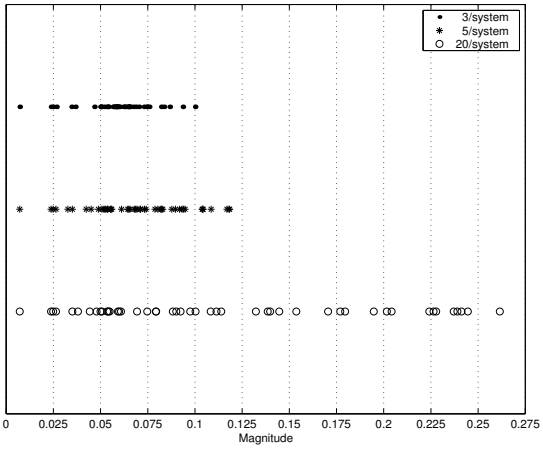


Figure 5.16: Magnitude of shifted eigenvalues using different start-up strategies to build a $M_{ISLRU(50)}$ preconditioner on the COBRA 60695 example.

increases monotonically with the number of shifted eigenvalues. The clear interest of this technique is shown in large parallel calculations for electromagnetics simulations. In this latter context, the solution technique enables us to reduce by a factor of up to eight the simulation times, that previously exceeded several hours of computation on a modern high performance computer.

For memory constraints which allow the storage of only a few eigenvectors, we investigate different strategies for the setup of the spectral preconditioner. If only a few right-hand sides have to be solved, a short setup phase consisting in collecting any candidate complying with the magnitude and the backward error conditions should be advised. If the sequence of the right-hand sides is large enough, a delaying strategy consisting in the selection of only a few candidates amongst the best ones should be preferred, as the resulting preconditioner will be more efficient in the stationary phase than the one built from a short setup.

The significant benefit of this approach is illustrated through extensive numerical experiments

but more theoretical studies would deserve to be undertaken to better understand the role of the sensitivity of the eigenvectors on the efficiency of the solver. This would enable us to better understand how this technique can be adapted in a nonlinear context where the matrix associated with the sequence of the right-hand sides varies slightly from one nonlinear step to the next. We think that extending the work developed for the symmetric Hermitian case [37] to the nonsymmetric situation can be a first track to follow.

Conclusions and prospectives

In this thesis, we have presented some preconditioning techniques suitable for the solution of sequences of right-hand sides. To assess the robustness and the efficiency of the preconditioning techniques, we studied the behaviour of the proposed techniques on challenging problems arising in large industrial calculations in electromagnetism.

Starting from ad-hoc application-dependent preconditioners, such as M_{Frob} for the electromagnetism calculation, we proposed techniques to improve their robustness. We first considered an improvement that consists in using the preconditioner to define a stationary scheme that eventually defines a new preconditioner M_{Iter} . A more sophisticated improvement technique M_{SLRU} implements a low-rank correction applied to the ad-hoc first level preconditioner. A possible weakness of this approach is that the spectral information is computed in a preprocessing phase using an eigensolver. If only a few right-hand sides have to be solved, this extra cost might not be amortized. To address this drawback, we used a variant of GMRES, namely GMRES-DR that solves the linear system and computes an approximation of the spectral information required by M_{SLRU} . When applied to a sequence of linear systems we proposed performing a series of updates using the selected spectral information from each linear solve for the subsequent one. This led us to define an incremental spectral preconditioner M_{ISLRU} for which we investigated some possible strategies to control the memory required to store the resulting preconditioner. Using our preconditioning techniques, we successfully accelerated significantly the calculation of a complete radar cross section calculation for objects of very large size.

For huge calculations, where M_{Frob} becomes less and less effective so that the use of an embedded scheme is the only way to solve the resulting linear systems, we think that using the incremental preconditioner on top of M_{iter} might be a fruitful track to follow. The incremental technique could be implemented as described in this manuscript between the solution of the sequence of right-hand sides. When only one right-hand side needs to be solved, we can think of implementing this technique to update the preconditioner used within the inner solves involved in an iterative embedded scheme. For example, for a Flexible-GMRES method, a GMRES-DR method combined with a spectral preconditioner can play the role of the flexible preconditioner. An improvement in the number of outer iterations could be expected if spectral information is recycled from one preconditioning operation to the next.

We mention that the calculation of radar cross section is a particular case of sequences of linear systems. Indeed, the right-hand sides are given simultaneously and better techniques based on block-Krylov solvers exist to deal with this situation [53]. For very large objects, the number of right-hand sides necessary to compute a radar cross section accurately becomes so large that its computation cannot be performed by a single block solver. A treatment based on sequences of blocks of right-hand sides must be considered. In that framework, we think that extending the incremental spectral preconditioner to the block situation deserves to be investigated. The block solver of choice is the block variant of GMRES-DR recently introduced in [65].

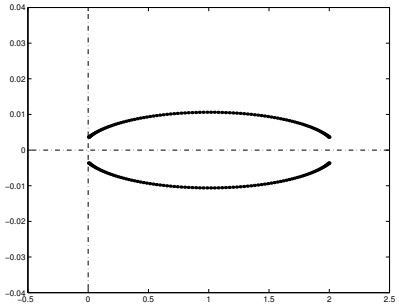
Finally, among the applications that could benefit from our work, still in electromagnetism, we point out the solution scheme where boundary elements and finite-elements are coupled. The overall solution is computed by alternatively solving the linear system associated with the finite-element discretization and the linear system associated with the boundary-element discretization. In that

context, we end up with two sequences of linear systems to be solved within a overall iterative scheme; the solution of both sequences of linear systems would benefit from our approach.

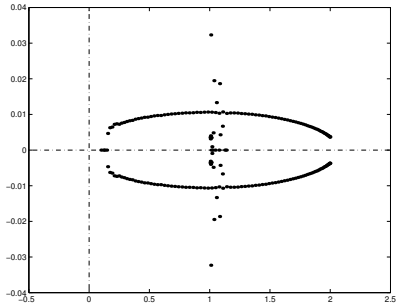
Appendix A

Effect of the low-rank updates on the spectrum of AM_{ILU}

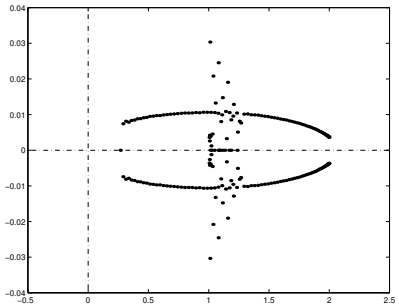
This Appendix is dedicated to illustrating the effect of the incremental spectral preconditioner $M_{ISLRU(k)}$ on the eigenvalue distribution of the preconditioned matrix AM_{ILU}, when varying the number k of shifted eigenvalues as described in Chapter 5. We consider a sequence of 31 right-hand sides defined by $\alpha = 10^{-1}$ (Equation (5.7)) with the initial guess strategy described in Section 5.4.2, and all the test matrices shown in Section 5.1. For each test case, we draw spectra of $AM_{ISLRU(k)}$ obtained for different systems in the sequence. As we use inexact eigeninformation, the unwanted eigenvalues are not exactly shifted to one while the rest of the spectrum is unchanged. Applying a spectral low-rank update to the preconditioner has an impact on the whole eigenvalue distribution in most of the cases. However, as can be seen in the plots, the incremental spectral preconditioner succeeds in recovering eigenvalues lying in the neighbourhood of zero and in clustering more eigenvalues to one.



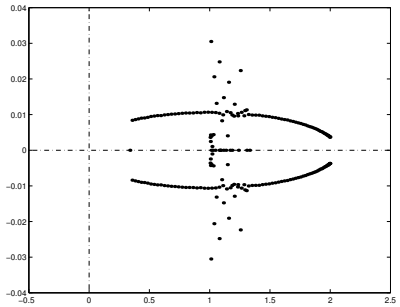
(a) 1^{-st} Syst – k = 0



(b) 10^{-th} Syst – k = 30

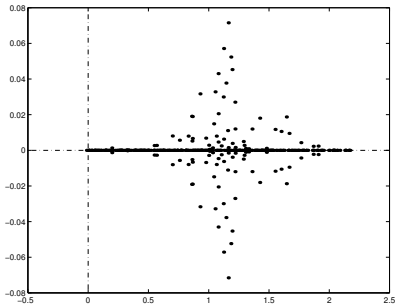


(c) 20^{-th} Syst – k = 47

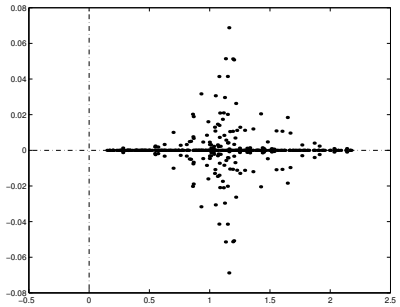


(d) 31^{-th} Syst – k = 53

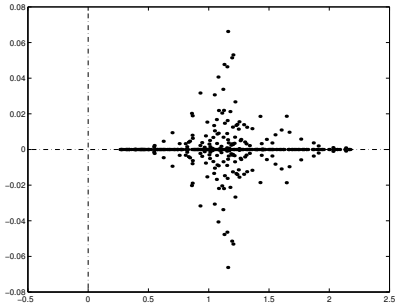
Figure A.1: Spectrum of $AM_{ISLRU(k)}$ for different systems of the sequence, where A is the matrix BWM200.



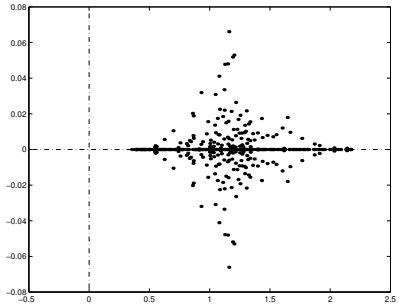
(a) 1^{-st} Syst – k = 0



(b) 10^{-th} Syst – k = 30

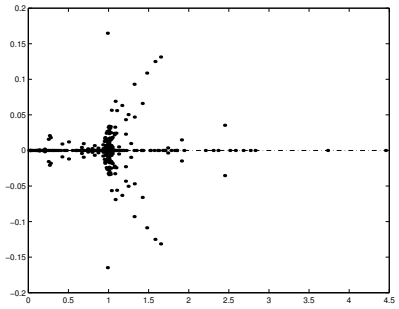


(c) 20^{-th} Syst – k = 46

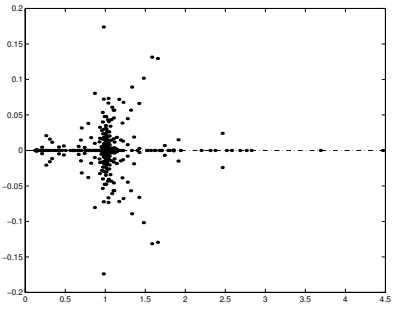


(d) 31^{-th} Syst – k = 62

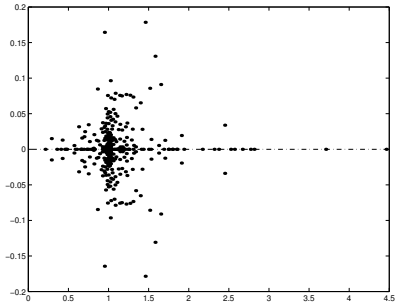
Figure A.2: Spectrum of $AM_{ISLRU(k)}$ for different systems of the sequence, where A is the matrix BFW398A.



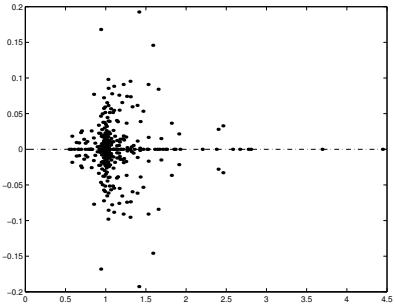
(a) 1^{-st} Syst – k = 0



(b) 5^{-th} Syst – k = 20

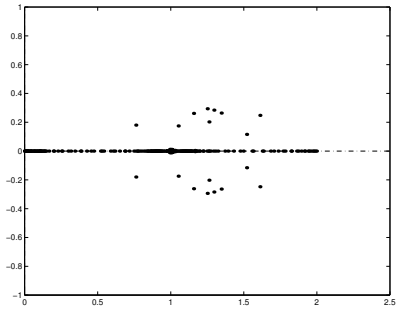


(c) 10^{-th} Syst – k = 44

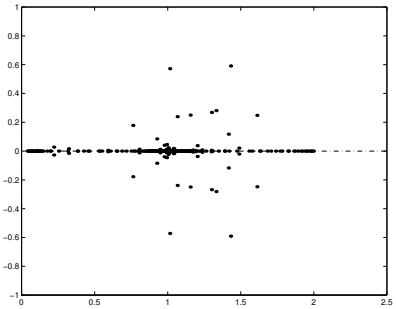


(d) 14^{-th} Syst – k = 54

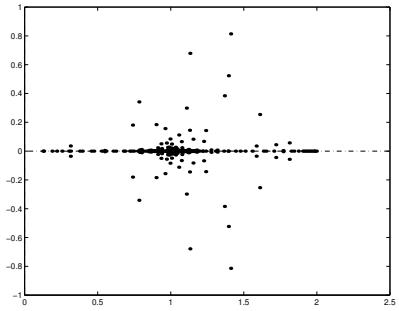
Figure A.3: Spectrum of $AM_{ISLRU(k)}$ for different systems of the sequence, where A is the matrix HOR131.



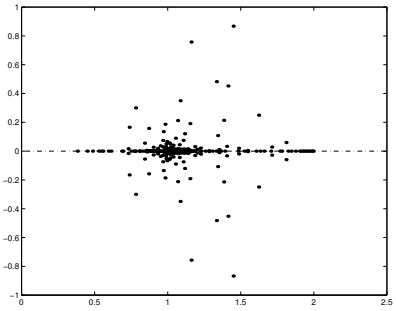
(a) 1^{-st} Syst – k = 0



(b) 5^{-th} Syst – k = 19

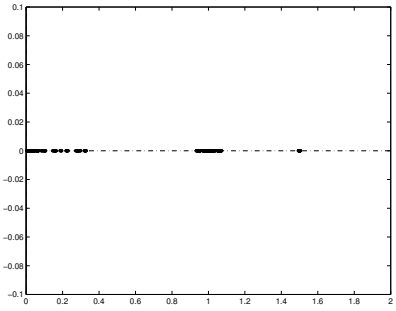


(c) 10^{-th} Syst – k = 44

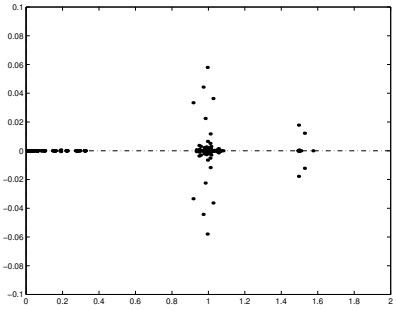


(d) 15^{-th} Syst – k = 55

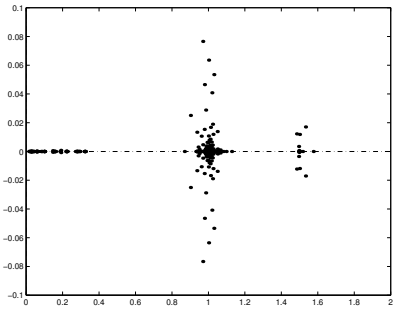
Figure A.4: Spectrum of $AM_{ISLRU(k)}$ for different systems of the sequence, where A is the matrix PORES3.



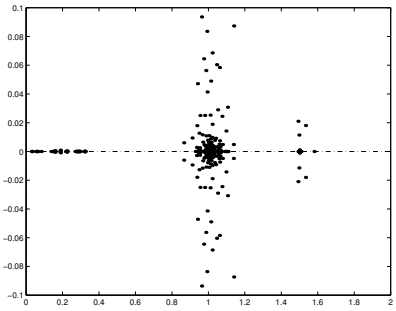
(a) 1^{-st} Syst – k = 0



(b) 10^{-th} Syst – k = 45

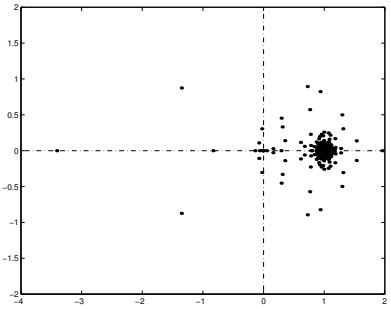


(c) 20^{-th} Syst – k = 95

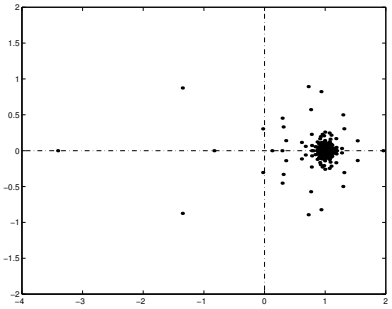


(d) 31^{-th} Syst – k = 148

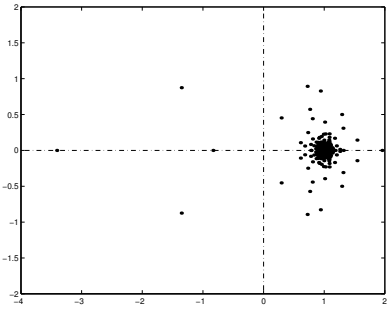
Figure A.5: Spectrum of $AM_{ISLRU(k)}$ for different systems of the sequence, where A is the matrix ORSIRR1.



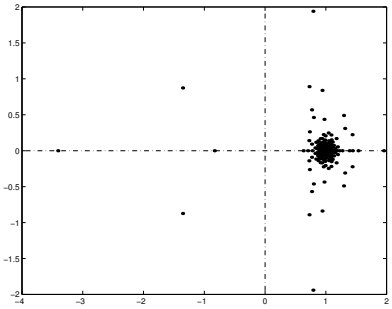
(a) 1^{-st} Syst – k = 0



(b) 3^{-th} Syst – k = 9

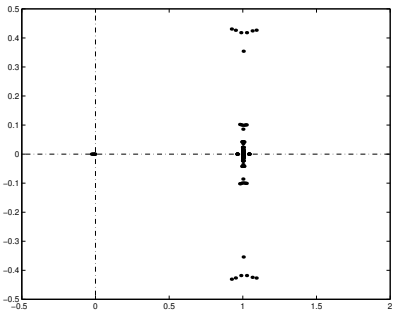


(c) 5^{-th} Syst – k = 15

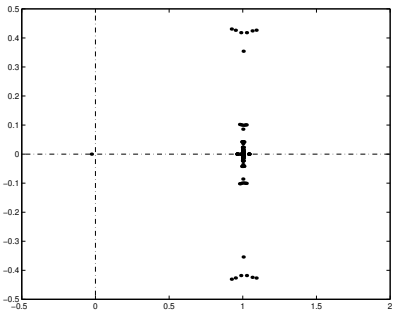


(d) 6^{-th} Syst – k = 17

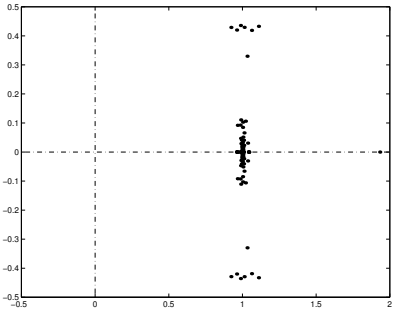
Figure A.6: Spectrum of $AM_{ISLRU(k)}$ for different systems of the sequence, where A is the matrix GRE1107.



(a) 1^{-st} Syst – k = 0

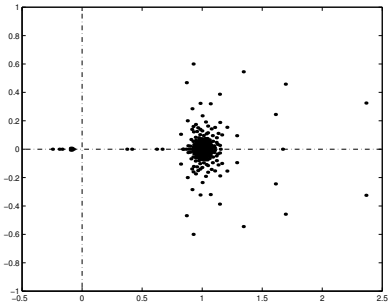


(b) 2^{-nd} Syst – k = 5

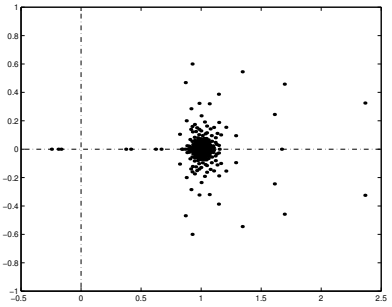


(c) 3^{-th} Syst – k = 7

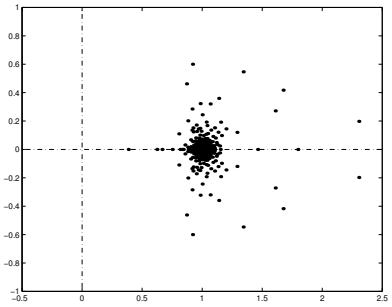
Figure A.7: Spectrum of $AM_{ISLRU(k)}$ for different systems of the sequence, where A is the matrix BWM2000.



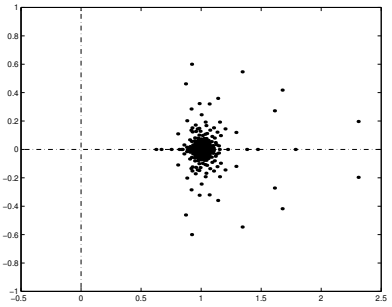
(a) 1^{-st} Syst – k = 0



(b) 2^{-nd} Syst – k = 5



(c) 3^{-th} Syst – k = 9



(d) 5^{-th} Syst – k = 12

Figure A.8: Spectrum of $AM_{ISLRU(k)}$ for different systems of the sequence, where A is the matrix RDB2048.

Bibliography

[1] 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.

[2] P. R. Amestoy, I. Duff, J. Y. L'Excellent, and J. Koster. MUltifrontal Massively Parallel Solver (MUMPS version 4.3): User's guide. URL : http://www.enseeiht.fr/lima/apo/MUMPS/, 2003.

[3] P. R. Amestoy, I. S. Duff, J. Koster, and J. Y. L'Excellent. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM J. Matrix Analysis and Applications*, 23:15–41, 2001.

[4] P. R. Amestoy, I. S. Duff, and J. Y. L'Excellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Comput. Methods in Appl. Mech. Eng.*, 184:501–520, 1998.

[5] M. Arioli, V. Pták, and Z. Strakoš. Krylov sequences of maximal length and convergence of GMRES. *BIT*, 38:636–643, 1998.

[6] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quaterly of applied mathematics*, 9:17–29, 1951.

[7] J. Baglama, D. Calvetti, G. H. Golub, and L. Reichel. Adaptively preconditioned GMRES algorithms. *SIAM J. Scientific Computing*, 20(1):243–269, 1999.

[8] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM Book, Philadelphia, PA, second edition, 1994.

[9] A. Bendali. *Approximation par elements finis de surface de problemes de diffraction des ondes electro-magnetiques*. PhD thesis, *Université Paris VI* , 1984.

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

[11] M. Benzi, C. D. Meyer, and M. Tůma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM J. Scientific Computing*, 17:1135–1149, 1996.

[12] M. Benzi and M. Tůma. A sparse approximate inverse preconditioner for nonsymmetric linear systems. *SIAM J. Scientific Computing*, 19:968–994, 1998.

[13] Å. Björck. Solving linear least squares problems by Gram-Schmidt orthogonalization. *BIT*, 7:1–21, 1967.

[14] 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.

[15] C. Le Calvez and B. Molina. Implicitly restarted and deflated GMRES. *Numerical Algorithms*, 21:261–285, 1999.

[16] B. Carpentieri. *Sparse preconditioners for dense linear systems from electromagnetic applications*. PhD thesis, CERFACS, Toulouse, France, 2002.

[17] 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*, 7(7-8):667–685, 2000.

[18] B. Carpentieri, I. S. Duff, and L. Giraud. A class of spectral two-level preconditioners. *SIAM J. Scientific Computing*, 25(2):749–765, 2003.

[19] B. Carpentieri, I. S. Duff, L. Giraud, and M. Magolu monga Made. Sparse symmetric preconditioners for dense linear systems in electromagnetism. *Numerical Linear Algebra with Applications*, 11(8-9):753–771, 2004.

[20] B. Carpentieri, I. S. Duff, L. Giraud, and G. Sylvand. Combining fast multipole techniques and an approximate inverse preconditioner for large parallel electromagnetics calculations. Technical Report TR/PA/03/77, CERFACS, Toulouse, France, 2003.

[21] B. Carpentieri, L. Giraud, and S. Gratton. Additive and multiplicative two-level spectral preconditioning for general linear systems. Technical Report TR/PA/04/38, CERFACS, Toulouse, France, 2004.

[22] F. Chaitin-Chatelin and V. Frayssé. *Lectures on Finite Precision Computations*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996. SIAM series Software · Environments · Tools, Editor in chief Jack J. Dongarra.

[23] K. Chen. An analysis of sparse approximate inverse preconditioners for boundary integral equations. *SIAM J. Matrix Analysis and Applications*, 22(3):1058–1078, 2001.

[24] D. Colton and R. Kress. *Integral Equation Methods in Scattering*. Wiley & Sons(New York), 1983.

[25] D. Colton and R. Kress. *Inverse Acoustic Electromagnetic Scattering Theory*. Applied Mathematical Science. Springer-Verlag, 1992.

[26] E. Darve. The fast multipole method (i) : Error analysis and asymptotic complexity. *SIAM J. Numerical Analysis*, 38(1):98–128, 2000.

[27] E. Darve. The fast multipole method: Numerical implementation. *J. Comp. Phys.*, 160(1):195–240, 2000.

[28] E. de Sturler. Inner-outer methods with deflation for linear systems with multiple right-hand sides. In *Householder Symposium XIII, Proceedings of the Householder Symposium on Numerical Algebra, Pontresina, Switzerland*, pages 193–196, June 17 - 26, 1996.

[29] J. Erhel, K. Burrage, and B. Pohl. Restarted GMRES preconditioned by deflation. *J. Comput. Appl. Math.*, 69:303–318, 1996.

[30] L. Fournier and S. Lanteri. Multiplicative and additive parallel multigrid algorithms for the acceleration of compressible flow computations on unstructured meshes. *Appl. Numer. Math.*, 36:401–426, 2001.

[31] J. Frank and C. Vuik. Parallel implementation of a multiblock method with approximate subdomain solution. *Appl. Num. Math.*, 30:403–423, 1999.

[32] V. Frayssé, L. Giraud, and S. Gratton. A set of Flexible-GMRES routines for real and complex arithmetics. Technical Report TR/PA/98/20, CERFACS, Toulouse, France, 1998.

[33] V. Frayssé, L. Giraud, S. Gratton, and J. Langou. A set of GMRES routines for real and complex arithmetics on high performance computers. Tech. Rep. TR/PA/03/03, CERFACS, 2003.

[34] V. Frayssé, L. Giraud, and H. Kharraz–Aroussi. On the influence of the orthogonalization scheme on the parallel performance of GMRES. In *Proceedings of EuroPar'98*, volume 1470 of *Lecture Notes in Computer Science*, pages 751–762. Springer–Verlag, 1998. A preliminary version is available as CERFACS Technical Reports, TR/PA/98/07.

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

[36] 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(1–3):119–157, 1997.

[37] L. Giraud and S. Gratton. On the sensitivity of some spectral preconditioners. Technical Report TR/PA/04/1008, CERFACS, Toulouse, France, 2004.

[38] L. Giraud, J. Langou, M. Rozložník, and J. van den Eshof. Rounding error analysis of the classical Gram-Schmidt orthogonalization process. Technical Report TR/PA/04/77, CERFACS, Toulouse, France, 2004.

[39] L. Giraud, J. Langou, and G. Sylvand. On the parallel solution of large industrial wave propagation problems. Technical Report TR/PA/04/52, CERFACS, Toulouse, France, 2004. Preliminary version of the paper to appear in Journal of Computational Acoustics.

[40] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. The Johns Hopkins University Press, Baltimore, MD, USA, third edition, 1996.

[41] A. Grama, V. Kumar, and A. Sameh. Parallel matrix-vector product using approximate hierarchical methods. In Sidney Karin, editor, *Proceedings of the 1995 ACM/IEEE Supercomputing Conference, December 3–8, 1995, San Diego Convention Center, San Diego, CA, USA*, New York, NY, USA, 1995. ACM Press and IEEE Computer Society Press.

[42] A. Greenbaum, V. Pták, and Z. Strakoš. Any nonincreasing convergence curve is possible for GMRES. *SIAM J. Matrix Analysis and Applications*, 17:465–469, 1996.

[43] L. Greengard and W. Gropp. A parallel version of the fast multipole method. *Comput. Math. Appl.*, 20:63–71, 1990.

[44] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73:325–348, 1987.

[45] M. Grote and T. Huckle. Parallel preconditionings with sparse approximate inverses. *SIAM J. Scientific Computing*, 18:838–853, 1997.

[46] W. Hackbusch. *Multigrid methods and applications*. Springer-Verlag, 1985.

[47] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, PA, USA, second edition, 2002.

[48] G. Karypis and V. Kumar. METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices - Version 4.0. University of Minnesota, 1998.

[49] 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.

[50] L. Yu. Kolotilina. Twofold deflation preconditioning of linear algebraic systems. I. Theory. Technical Report EM-RR 20/95, Elegant Mathematics, Inc., 1995. Available at: http://www.elegant-math.com/abs-emrr.htm.

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

[52] L. Yu Kolotilina and A. Yu. Yeremin. Factorized sparse approximate inverse preconditionings. II: Solution of 3D FE systems on massively parallel computers. *Int J. High Speed Computing*, 7:191–215, 1995.

[53] J. Langou. *Iterative methods for solving linear systems with multiple right hand sides.* Ph.D. dissertation, INSA Toulouse, June 2003. TH/PA/03/24.

[54] J. Lee, J. Zhang, and C. C. Lu. Sparse inverse preconditioning of multilevel fast multipole algorithm for hybrid integral equations in electromagnetics. Technical Report 363-02, Department of Computer Science, University of Kentucky, KY, 2002.

[55] J. Lee, J. Zhang, and C. C. Lu. Incomplete LU preconditioning for large scale dense complex linear systems from electromagnetic wave scattering problems. *J. Comp. Phys.*, 185:158–175, 2003.

[56] R. B. Lehoucq and A. G. Salinger. Large-scale eigenvalue calculations for stability analysis of steady flows on massively parallel computers. *Int. J. Numerical Methods in Fluids*, 36:309–327, 2001.

[57] R. B. Lehoucq and D. C. Sorensen. Deflation techniques for an implicitly restarted arnoldi iteration. *SIMAX*, 17:789–821, 1996.

[58] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK User's Guide: Solution of large-scale problem with implicitly restarted Arnoldi methods.* SIAM, Philadelphia, 1998.

[59] L. Mansfield. On the use of deflation to improve the convergence of conjugate gradient iteration. *Communs. Appl. Numer. Meth.*, 4:151–156, 1988.

[60] L. Mansfield. Damped Jacobi preconditioning and coarse grid deflation for conjugate gradient iteration on parallel computers. *SIAM J. Scientific and Statistical Computing*, 12:1314–1323, 1991.

[61] R. B. Morgan. A restarted GMRES method augmented with eigenvectors. *SIAM J. Matrix Analysis and Applications*, 16:1154–1171, 1995.

[62] R. B. Morgan. Harmonic projection methods for large non-symmetric eigenvalue problems. *Numerical Linear Algebra with Applications*, 5:33–55, 1998.

[63] R. B. Morgan. Implicitly restarted GMRES and Arnoldi methods for nonsymmetric systems of equations. *SIAM J. Matrix Analysis and Applications*, 21(4):1112–1135, 2000.

[64] R. B. Morgan. GMRES with deflated restarting. *SIAM J. Scientific Computing*, 24(1):20–37, 2002.

[65] R. B. Morgan. Restarted block GMRES with deflation of eigenvalues. *Applied Numerical Mathematics*, to appear.

[66] R. Nicolaides. Deflation of conjugate gradients with applications to boundary value problems. *SIAM J. Numerical Analysis*, 24:355–365, 1987.

[67] D. P. O'Leary. The block conjugate gradient algorithm and related methods. *Linear Algebra and its Applications*, 29:293–322, 1980.

[68] M. L. Parks, E. de Sturler, G. Mackey, D. D. Jhonson, and S. Maiti. Recycling Krylov subspaces for sequences of linear systems. Technical Report UIUCDCS-R-2004-2421 (CS), University of Illinois at Urbana-Champaign, 2004.

[69] P. A. Raviart and J. M. Thomas. A mixed finite element method for 2nd order elliptic problems. In I. Galligani and E. Magenes, editors, *Mathematical aspects of finite element method*, volume 606 of *Lecture Notes in Mathematics*. Springer–Verlag, Berlin, 1975.

[70] J. L. Rigal and J. Gaches. On the compatibility of a given solution with the data of a linear system. *J. Assoc. Comput. Mach.*, 14(3):543–548, 1967.

[71] Y. Saad. Projection and deflation methods for partial pole assignment in linear state feedback. *IEEE Trans. Automat. Contr.*, 33(3):290–297, 1988.

[72] Y. Saad. *Numerical methods for large eigenvalue problems*. Halsted Press, New York, NY, 1992.

[73] Y. Saad. Analysis of augmented Krylov subspace techniques. *SIAM J. Scientific Computing*, 14:461–469, 1993.

[74] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Scientific and Statistical Computing*, 14:461–469, 1993.

[75] Y. Saad. ILUT: A dual threshold incomplete LU factorization. *Numerical Linear Algebra with Applications*, 1:387–402, 1994.

[76] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Scientific and Statistical Computing*, 7:856–869, 1986.

[77] A. R. Samant, E. Michielssen, and P. Saylor. Approximate inverse based preconditioners for 2d dense matrix problems. Technical Report CCEM-11-96, University of Illinois, 1996.

[78] K. Sertel and J. L. Volakis. Incomplete LU preconditioner for FMM implementation. *Microwave and Optical Technology Letters*, 26(7):265–267, 2000.

[79] J. N. Shadid and R. S. Tuminaro. A comparison of preconditioned nonsymmetric Krylov methods on a large-scale MIMD machine. *SIAM J. Scientific Computing*, 14(2):440–459, 1994.

[80] V. Simoncini and E. Gallopoulos. An iterative method for nonsymmetric systems with multiple right-hand sides. *SIAM J. Scientific Computing*, 16:917–933, 1995.

[81] B. F. Smith, P. E. Bjørstad, and W. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1996.

[82] J. M. Song, C. C. Lu, and W. C. Chew. Multilevel fast multipole algorithm for electromagnetic scattering. *IEEE Antennas and Propagation Magazine*, 45:1488–1493, 1997.

[83] D. C. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Analysis and Applications*, 13:357–385, 1992.

[84] G. W. Stewart and J. G. Sun. *Matrix Perturbation Theory*. Academic Press, New York, 1991.

[85] G. Sylvand. *La méthode multipôle rapide en électromagnétisme : performances, parallélisation, applications*. PhD thesis, Ecole Nationale des Ponts et Chaussées, 2002.

[86] R. S. Tuminaro. A highly parallel multigrid-like method for the solution of the euler equations. *SIAM J. Scientific and Statistical Computing*, 13:88–100, 1992.

[87] S. A. Vavasis. Preconditioning for boundary integral equations. *SIAM J. Matrix Analysis and Applications*, 13:905–925, 1992.

[88] B. Vital. *Étude de quelques méthodes de résolution de problèmes linéaires de grande taille sur multiprocesseur*. Ph.D. dissertation, Université de Rennes, November 1990.

[89] J. H. Wilkinson. *Rounding Errors in Algebraic Processes*. London, 1963.

[90] F. Zhao and S. L. Johnsson. The parallel multipole method on the connection machine. *SIAM J. Scientific and Statistical Computing*, 12:1420–1437, 1991.