On the parallel solution of large industrial wave propagation problems

L. Giraud [*] J. Langou [†] G. Sylvand [‡]

CERFACS Technical Report TR/PA/04/52 - June 2004

Abstract

The use of Fast Multipole Methods (FMM) combined with embedded Krylov solvers preconditioned by a sparse approximate inverse is investigated for the solution of large linear systems arising in industrial acoustic and electromagnetic simulations. We use a boundary elements integral equation method to solve the Helmholtz and the Maxwell equations in the frequency domain. The resulting linear systems are solved by iterative solvers using FMM to accelerate the matrix-vector products. The simulation code is developed in a distributed memory environment using message passing and it has out-of-core capabilities to handle very large calculations. When the calculation involves one incident wave, one linear system has to be solved. In this situation, embedded solvers can be combined with approximate inverse preconditioner to design extremely robust algorithms. For radar cross section calculations, several linear systems have to be solved. They involve the same coefficient matrix but different right-hand sides. In this case, we propose a block variant of the single right-hand side scheme. The efficiency, robustness and parallel scalability of our approach are illustrated on a set of large academic and industrial test problems.

Keywords: Large electromagnetic and acoustic simulations, high performance computing, distributed memory environment, fast multipole methods, flexible Krylov solvers, relaxed inner-outer schemes, approximate inverse preconditioner.

1 Introduction

In the last decade, a significant amount of effort has been spent on the simulation of wave propagation phenomena to address various topics in research or engineering applications. Two complementary approaches based on the solution of the wave propagation equations are often adopted for tackling these problems. The first approach utilizes finite differencing in a volumic formulation. The second one operates with a surfacic formulation. The latter offers two main advantages: (a) it does not require truncating the infinite spatial domain surrounding the scatterer and using approximate boundary conditions, and (b) it requires discretizing only the surface of the scatterer. On the other hand, the boundary elements approach leads to singular integral equations of the first kind, the discretization of which results in linear systems with complex and dense matrices which are quite challenging to solve. With the advent of parallel processing, both volumic and surfacic methods have witnessed important developments and the typical problem size in industry has increased by a few orders of magnitude.

Direct solution methods have been for years the method of choice for solving dense linear systems coming from boundary integral equations because they are reliable and predictable both

[*]CERFACS, 42 Avenue G. Coriolis, 31057 Toulouse Cedex, France. giraud@cerfacs.fr

[†]The University of Tennessee, Department of Computer Science, 1122 Volunteer Blvd., Knoxville, Tennessee, 37996-3450, USA. langou@cs.utk.edu

[‡]EADS CCR, Centre de Toulouse, Centreda 1, 4, Avenue Didier Daurat, 31700 Blagnac, France. guillaume.sylvand@eads.net

in terms of accuracy and computational cost. However, for the solution of large-scale problems, direct methods are infeasible even on large parallel platforms because they require an unfordable storage and an unfordable number of floating-point operations (if n is the problem size, then the storage grows as $\mathcal{O}(n^2)$ and the computational cost as $\mathcal{O}(n^3)$).

The use of preconditioned Krylov solvers can be an alternative to direct solution methods, provided we have fast matrix-free matrix-vector products and robust preconditioners. Research efforts have recently concentrated on fast methods for performing matrix-vector products with $\mathcal{O}(n\log(n))$ computational complexity, including strategies for parallel distributed memory implementations. These methods, generally referred to as *hierarchical methods*, were introduced originally in the context of the study of particle simulations and can be effectively used in boundary element applications.

In this paper, we focus on the design of a parallel distributed solver for wave propagation simulations. The solver combines parallel distributed preconditioned Krylov methods and fast multipole methods. The paper is organized as follows. In Section 2 we describe the underlying equations associated with electromagnetic and acoustic problems. We then present the numerical tools that have been combined to design the software. More precisely Section 3.1 is devoted to the Fast Multipole technique and its parallel implementation in a distributed memory environment while Section 3.2 presents the linear algebra components. In Section 4 we report on the numerous numerical experiments perform on various parallel platforms. We illustrate on a set of large problems the efficiency of the solver as well as its parallel performance.

2 The integral equation formulation and its discretizations

In this section, we introduce the integral equations used in our study, first in acoustic then in electromagnetism.

2.1 Helmholtz equation

2.1.1 Acoustic problem

We are interested in the solution of the Helmholtz equations in the frequency domain using an integral equation formulation. We consider the case of an object Ω^- illuminated by an incident plane wave u_{inc} of pulsation ω and wave number $k = \omega/c$ where c is the wave celerity. The boundary Γ of the object has a rigid part Γ_r and a treated part Γ_t of impedance η. The outer normal is ν.

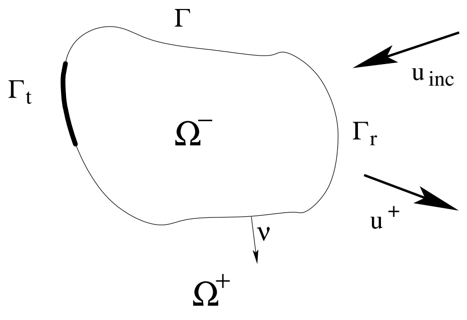


Figure 1: Acoustic problem

We want to compute the diffracted field u^+ in the outer domain Ω^+. It is solution of the problem (P^+) :

$$(P^+)\begin{cases}\Delta u^+ + k^2 u^+ = 0 & \textit{in } \Omega^+,\\ \dfrac{\partial u^+}{\partial \nu} = -\dfrac{\partial u_{inc}}{\partial \nu} & \textit{on } \Gamma_r,\\ \dfrac{\partial u^+}{\partial \nu} + i\dfrac{k}{\eta}u^+ = -\left(\dfrac{\partial u_{inc}}{\partial \nu} + i\dfrac{k}{\eta}u_{inc}\right) & \textit{on } \Gamma_t,\\ \lim\limits_{r\to+\infty} r\left(\dfrac{\partial u^+}{\partial r} - iku^+\right) = 0. \end{cases}$$

We associate to (P^+) the problem (P^-) in the inner domain :

$$(P^-)\begin{cases}\Delta u^- + k^2 u^- = 0 & \textit{in } \Omega^-,\\ \dfrac{\partial u^-}{\partial \nu} = -\dfrac{\partial u_{inc}}{\partial \nu} & \textit{on } \Gamma_r,\\ \dfrac{\partial u^-}{\partial \nu} - i\dfrac{k}{\eta}u^- = -\left(\dfrac{\partial u_{inc}}{\partial \nu} - i\dfrac{k}{\eta}u_{inc}\right) & \textit{on } \Gamma_t. \end{cases}$$

2.1.2 Integral formulation

We now denote ϕ and p the jumps of the tangential parts of u and $\partial u/\partial\nu$ on Γ :

$$\begin{cases}\phi(x) = u^-_{|\Gamma}(x) - u^+_{|\Gamma}(x) & x\in\Gamma,\\ p(x) = \left(\dfrac{\partial u^-}{\partial \nu}\right)_{|\Gamma}(x) - \left(\dfrac{\partial u^+}{\partial \nu}\right)_{|\Gamma}(x) & x\in\Gamma.\end{cases}$$

We know that $\phi \in H^{1/2}_\Gamma$ and $p \in H^{-1/2}_\Gamma$. The knowledge of p and ϕ entirely solves the problem, since the following representation theorem allow then to compute the diffracted pressure u in any point x (see [15]) :

$$u(x) = \int_\Gamma \left(G(x,y)p(y) - \frac{\partial G(x,y)}{\partial \nu_y}\phi(y)\right)dy,$$

where $G(x,y)$ is the Green's function, solution of $\Delta u + k^2 u = -\delta_0$, that is

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

For homogenity reason, we replace the variable p by $\lambda = -p/ik$. We then obtain the variationnal formulation : find (ϕ,λ) such that $\forall(\phi^t,\lambda^t)$, we have

$$\begin{cases} -\displaystyle\int_{\Gamma_r\times\Gamma_t}\frac{\partial G}{\partial\nu_x}\lambda(y)\phi^t(x)dydx - \frac{1}{ik}\oint_{\Gamma_r\times\Gamma}\frac{\partial^2 G}{\partial\nu_x\partial\nu_y}\phi(y)\phi^t(x)dydx\\ \qquad = -\displaystyle\frac{1}{ik}\int_{\Gamma_r}\frac{\partial u_{inc}(x)}{\partial\nu}\phi^t(x)dx,\\ -\displaystyle\frac{1}{ik}\oint_{\Gamma_t\times\Gamma}\frac{\partial^2 G}{\partial\nu_x\partial\nu_y}\phi(y)\phi^t(x)dydx - \frac{1}{2\eta}\int_{\Gamma_t}\phi(x)\phi^t(x)dx\\ \qquad -\displaystyle\int_{\Gamma_t\times\Gamma_t}\frac{\partial G}{\partial\nu_x}\lambda(y)\phi^t(x)dydx = -\frac{1}{ik}\int_{\Gamma_t}\frac{\partial u_{inc}(x)}{\partial\nu}\phi^t(x)dx,\\ \displaystyle\frac{\eta}{2}\int_{\Gamma_t}\lambda(x)\lambda^t(x)dx - ik\int_{\Gamma_t\times\Gamma_t}G\lambda(y)\lambda^t(x)dydx - \int_{\Gamma_t\times\Gamma}\frac{\partial G}{\partial\nu_y}\phi(y)\lambda^t(x)dydx\\ \qquad = -\displaystyle\int_{\Gamma_t}u_{inc}(x)\lambda^t(x)dx.\end{cases} \tag{1}$$

In the case of a rigid body, all the terms involving Γ_t, η, λ^t and λ disappear. The system (1) becomes

$$-\frac{1}{ik}\oint_{\Gamma\times\Gamma}\frac{\partial^2 G}{\partial\nu_x\partial\nu_y}\phi(y)\phi^t(x)dydx = -\frac{1}{ik}\int_\Gamma\frac{\partial u_{inc}(x)}{\partial\nu}\phi^t(x)dx. \tag{2}$$

2.1.3 Discretization

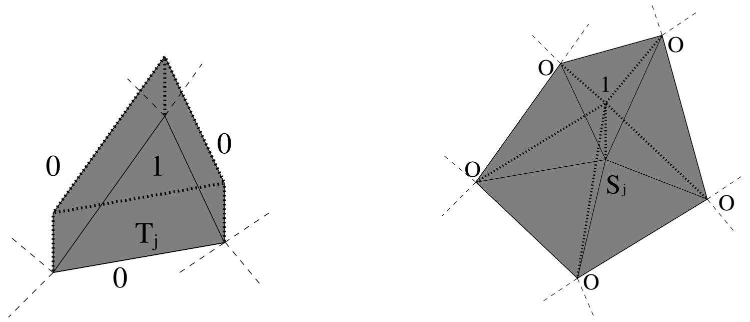


Figure 2: P0 (left) and P1 (right) basis functions

In order to discretize this system, we use a surfacic triangle mesh of the boundary Γ. The pressure jump ϕ is discretized using P1 linear basis functions, the pressure normal derivative jump λ is discretized using P0 basis functions (cf. Figure 2). We end-up with a complex, dense, symmetric linear system to solve.

2.2 Maxwell equations

2.2.1 Problem definition

We are now interested in the case of an object Ω_i of frontier Γ made of a dielectric material caracterized by its electric permittivity ϵ_i and magnetic permeability μ_i. The quantities ϵ_i and μ_i can be complex, in the case of absorbing material. The wave speed in this media $c_i = 1/\sqrt{\epsilon_i\mu_i}$, the wave length $\lambda_i = c_i/f$ (f being the frequency), the wave number is $k_i = 2\pi/\lambda_i$, the associated Green's function is $G_i(r) = e^{ik_i r}/4\pi r$. We denote Ω_e the complementary of Ω_i, and ϵ_e, μ_e, c_e, λ_e, k_e, G_e the corresponding data. The outgoing normal on Γ is denoted ν. Ω is illuminated by an incident plane wave $(\vec{E}_{inc}, \vec{H}_{inc})$.

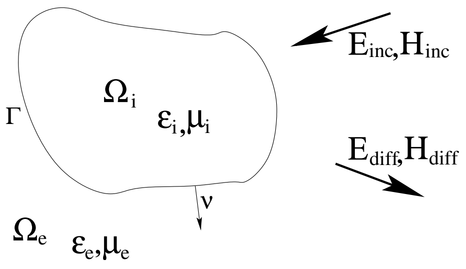


Figure 3: Maxwell problem

In Ω_e, the diffracted fields are solution of :

$$(P_e) \left\{ \begin{array}{l} \vec{rot}\vec{E}_e - i\omega\mu_e\vec{H}_e = 0, \\ \vec{rot}\vec{H}_e + i\omega\epsilon_e\vec{E}_e = 0, \\ \lim\limits_{r\to+\infty} r\left|\sqrt{\epsilon_e}\vec{E}_e - \sqrt{\mu_e}\vec{H}_e \wedge \frac{\vec{r}}{r}\right| = 0 \end{array} \right. \quad in\ \Omega_e. \tag{3}$$

In Ω_i, the unknowns are the total fields, they are given by :

$$(P_i) \left\{ \begin{array}{l} \vec{rot}\vec{E}_i - i\omega\mu_i\vec{H}_i = 0, \\ \vec{rot}\vec{H}_i + i\omega\epsilon_i\vec{E}_i = 0 \end{array} \right. \quad in\ \Omega_i. \tag{4}$$

Finally, in order to link (P_i) and (P_e), we must write the continuity of the tangential parts of the total fields $\vec{E}$ and $\vec{H}$ on the boundary Γ

$$\left\{ \begin{array}{l} \vec{E}_i \wedge \vec{\nu} = (\vec{E}_e + \vec{E}_{inc}) \wedge \vec{\nu}, \\ \vec{H}_i \wedge \vec{\nu} = (\vec{H}_e + \vec{H}_{inc}) \wedge \vec{\nu}. \end{array} \right. \quad in\ \Gamma. \tag{5}$$

We now write an integral formulation in order to solve the problem defined by (3), (4) and (5).

2.2.2 Integral equation

We introduce the electric and magnetic surfacic currents $\vec{j}$ and $\vec{m}$ defined on Γ as the tangential parts of the total fields $\vec{H}$ and $\vec{E}$:

$$\left\{ \begin{array}{l} \vec{j} = \vec{\nu} \wedge \vec{H}_{tot}, \\ \vec{m} = \vec{\nu} \wedge \vec{E}_{tot} \end{array} \right. \quad on\ \Gamma.$$

We introduce the operators R and S defined by :

$$\left\{ \begin{array}{l} \vec{Rj}(y) = \int_\Gamma \vec{grad}_y G(|y-x|) \wedge \vec{j}(x)dx, \\ \vec{Sj}(y) = \int_\Gamma G(|y-x|)\vec{j}(x) + \frac{1}{k^2}\vec{grad}_y G(|y-x|) div_\Gamma \vec{j}(x)dx. \end{array} \right.$$

Maxwell representation theorem allows to compute the fields $\vec{H}$ and $\vec{E}$ at any point $y \notin \Gamma$

$$\left\{ \begin{array}{l} \vec{E}(y) = \vec{Rm}(y) + i\omega\mu\vec{Sj}(y), \\ \vec{H}(y) = \vec{Rj}(y) - i\omega\epsilon\vec{Sm}(y). \end{array} \right. \tag{6}$$

Therefore the knowledge of $\vec{j}$ and $\vec{m}$ solves the whole problem. Using (6) in both domain leads to the following equation on Γ :

$$\left\{ \begin{array}{l} (\vec{R_e m} + \vec{R_i m}) + i\omega(\mu_e\vec{S_e j} + \mu_i\vec{S_i j}) = -\vec{E}_{inc}, \\ (\vec{R_e j} + \vec{R_i j}) - i\omega(\epsilon_e\vec{S_e m} + \epsilon_i\vec{S_i m}) = -\vec{H}_{inc}. \end{array} \right.$$

Testing these equations with tangential test functions $(\vec{j}^t, \vec{m}^t)$, we obtain the variationnal formulation : find $(\vec{j}, \vec{m})$ such that $\forall\vec{j}^t$ and $\forall\vec{m}^t$, we have

$$\left\{ \begin{array}{l} \int_\Gamma(\vec{R_e m} + \vec{R_i m}).\vec{j}^t + i\omega\int_\Gamma(\mu_e\vec{S_e j} + \mu_i\vec{S_i j}).\vec{j}^t = -\int_\Gamma \vec{E}_{inc}.\vec{j}^t, \\ \int_\Gamma(\vec{R_e j} + \vec{R_i j}).\vec{m}^t - i\omega\int_\Gamma(\epsilon_e\vec{S_e m} + \epsilon_i\vec{S_i m}).\vec{m}^t = -\int_\Gamma \vec{H}_{inc}.\vec{m}^t. \end{array} \right. \tag{7}$$

In the case of a perfectly conducting object, $\vec{m}$ is null, $\vec{m}^t$ disappears and (7) becomes the following equation called EFIE (Electric Field Integral Equation) :

$$i\omega \int_\Gamma (\mu_e \vec{S_e j}).\vec{j}^t = -\int_\Gamma \vec{E}_{inc}.\vec{j}^t, \tag{8}$$

If the object is smooth, perfectly conducting and closed, we can also write the MFIE (Magnetic Field Integral Equation) :

$$\int_\Gamma (\vec{R_e j} \wedge \vec{\nu}).\vec{j}^t + \frac{1}{2}\int_\Gamma \vec{j}.\vec{j}^t = -\int_\Gamma (\vec{H}_{inc} \wedge \vec{\nu}).\vec{j}^t, \tag{9}$$

EFIE is symmetric, whereas MFIE is not. Finally, in order to remove the problem of inner resonnant frequencies, we can introduce the CFIE (Combined Field Integral Equation), which is a linear combination of EFIE (8) and MFIE (9) : $CFIE = \alpha EFIE + (1-\alpha)\frac{i}{k}MFIE$. Any choice for α is acceptable but $\alpha = 0.2$ usually gives good results. Hence, CFIE is given by

$$\int_\Gamma \left[\alpha(i\omega\mu_e \vec{S_e j}) + (1-\alpha)\frac{i}{k}(\vec{R_e j} \wedge \vec{\nu} + \frac{\vec{j}}{2})\right].\vec{j}^t = -\int_\Gamma \left[\alpha \vec{E}_{inc} + (1-\alpha)\frac{i}{k}\vec{H}_{inc} \wedge \vec{\nu}\right].\vec{j}^t. \tag{10}$$

2.2.3 Discretization

Once again, we have a surfacic triangle mesh of Γ. We use the standard discretization for this kind of problem, that of Raviart-Thomas [17] applied in the current context a few years later by Rao-Wilton-Glisson [16]. The basis functions are associated with the edges of the mesh. Both unknowns $\vec{j}$ and $\vec{m}$ use the same basis functions. It leads to a symmetric, dense, complex, non-hermitian matrix. In the case of a dielectric object, the number of unknowns is twice the number of edges.

3 The parallel numerical solution

3.1 The parallel Fast Multipole calculation

3.1.1 Introduction

The integral approach used here to solve the Helmholtz and Maxwell equations has several advantages over a classic surfacic formulation. First, it requires to mesh only the surface of the objects, not the propagation media inside and around it. In an industrial context, where the mesh preparation is a very time consuming step, this is crucial. Second, the radiation condition at infinity is treated naturally by the formulation in an exact manner. At last, the surfacic mesh allows to have a very accurate description of the object's shape, which is not the case with finite difference method for instance.

The main drawback of integral formulation is that it leads to solve dense linear systems difficult to tackle with iterative solvers. Even though the spectral condition number is usually not very high, the eigenvalues distribution of these matrices is not favorable to fast convergence of unpreconditioned Krylov solvers [1]. Furthermore for large objects and/or large frequencies, the number of unknowns can easily reach several millions. In that case, the classic iterative and direct solvers are unable to solve our problem, because they become very expensive in CPU time and storage requirements.

The Fast Multipole Method (FMM) is a new way to compute fast but approximate matrix-vector products. In conjonction with any iterative solver, it is an efficient way to overcome these limitations [25, 24, 22]. It is fast in the sense that CPU time is $\mathcal{O}(n.\log(n))$ instead of $\mathcal{O}(n^2)$ for standard matrix-vector products, and approximate in the sense that there is a relative error between the "old" and the "new" matrix-vector products $\varepsilon \approx 10^{-3}$. Nevertheless, this error is not a problem since it is usually below the target accuracy requested to the iterative solver or below the error introduced by the surfacic triangle approximation.

In the Helmholtz case, the input data of the FMM is a vector that represents surfacic potentials (ϕ, λ) and the aim of the FMM is to compute the left hand side of (1) for any test function ϕ^t or λ^t. In the Maxwell case, the input data of the FMM is a vector that represents surfacic currents $(\vec{j}, \vec{m})$ defined on Γ, and we mean to compute the left hand side of (7), (8), (9) or (10) for any test function $\vec{j}^t$ or $\vec{m}^t$.

3.1.2 The multipole algorithm

We give now an overview of the single level multipole algorithm. We first consider a simple equation: for a given surfacic function $\vec{j}$, and for any test function $\vec{j}^t$, we want to compute

$$\int_{\Gamma\times\Gamma} G(|y-x|)\vec{j}(x)\vec{j}^t(y)dydx. \tag{11}$$

We first split the surface Γ into equally sized domain using for instance a cubic grid (as shown in Figure 4). The degrees of freedom are then dispatched between these cubic cells. Interaction of basis functions located in neighbouring cells (that is, cells that share at least one vertex) are treated classically, without any multipole acceleration.

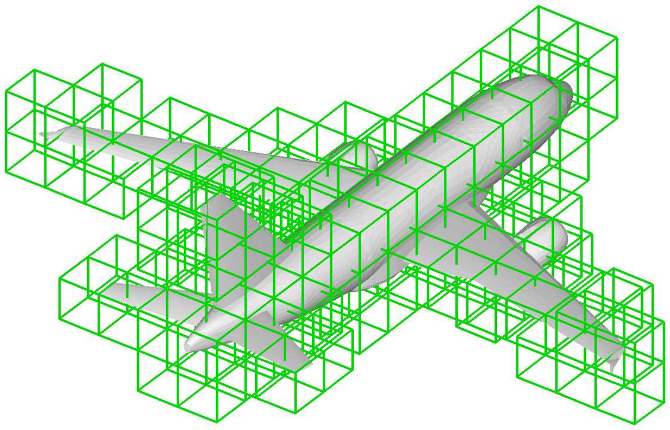


Figure 4: Use of a cubic grid to split the mesh

The calculation of the interactions between unknowns located in non-neighbouring cells are accelerated with the FMM. The base of this algorithm is the following addition theorem: given two points x and y located in two distant (=non-neighbouring) boxes $\mathcal{C}$ and $\mathcal{C}'$ centered in M and M', we have

$$G(|y-x|) = \frac{ik}{16\pi^2} \lim_{L\to+\infty} \int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.\vec{xM}} T^L_{\vec{MM'}}(\vec{s}) e^{ik\vec{s}.\vec{M'y}} d\vec{s}, \tag{12}$$

where $\mathcal{S}$ denotes the unit sphere in $\mathbb{R}^3$, and $T^L_{\vec{MM'}}$ is the transfer function defined on $\mathcal{S}$ by

$$T^L_{\vec{MM'}}(\vec{s}) = \sum_{0\le l\le L} (2l+1)i^l h_l^{(1)}(k.|\vec{MM'}|)P_l(cos(\vec{s}, \vec{MM'})). \tag{13}$$

The parameter L is called number of poles. It is chosen in accordance with the size of the box edge a, in order to have a good accuracy in (12) and no divergence in (13): $L = \sqrt{3}ka$ satisfies these two conditions.

Inserting the addition theorem (12) in the term to compute (11), we see that the term of interaction between $\Gamma \cap \mathcal{C}$ and $\Gamma \cap \mathcal{C}'$, which is equal to $\int_{\Gamma\cap\mathcal{C}\times\Gamma\cap\mathcal{C}'} G(|y-x|)\vec{j}(x)\vec{j}^t(y)dydx$, can now be computed in three steps.

- **Initialization:** we compute the function $\mathcal{F}_\mathcal{C}$ defined on the unit sphere $\mathcal{S}$
$$\mathcal{F}_\mathcal{C}(\vec{s}) = \int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.\vec{xM}}\vec{j}(x)dx, \tag{14}$$
$\mathcal{F}_\mathcal{C}$ depends only on the current $\vec{j}$, on the cell $\mathcal{C}$ and its center M. It represents the action of the current $\vec{j}(x)$ for $x \in \Gamma\cap\mathcal{C}$ on *any* distant cell $\mathcal{C}'$.

- **Transfer:** we multiply $\mathcal{F}_\mathcal{C}$ by the transfer function $T^L_{\vec{MM'}}$
$$\mathcal{G}_{\mathcal{C}'}(\vec{s}) = \mathcal{F}_\mathcal{C}(\vec{s})T^L_{\vec{MM'}}(\vec{s}). \tag{15}$$
The result $\mathcal{G}_{\mathcal{C}'}(\vec{s})$ is still a function defined on the unit sphere, and it represents the action of the current $\vec{j}(x)$ for $x \in \Gamma\cap\mathcal{C}$ *specifically* on $\mathcal{C}'$.

- **Integration:** we finish this calculus by integrating both on $\mathcal{S}$ and on $\Gamma\cap\mathcal{C}'$
$$\frac{ik}{16\pi^2}\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} \mathcal{G}_{\mathcal{C}'}(\vec{s}).e^{ik\vec{s}.\vec{M'y}}\vec{j}^t(y)d\vec{s}dy. \tag{16}$$

We obtain the action of $\mathcal{C}$ on $\mathcal{C}'$. By doing so for all the couples of non-neighbouring boxes, and by treating classically the interaction between neighbouring boxes, we can compute the entire matrix vector product. This algorithm is very technical, and has been extensively explained in the litterature [8, 10, 27]. The optimal complexity of the single level FMM is $\mathcal{O}(n_{dof}^{3/2})$ (where n_{dof} is the number of degrees of freedom) against $\mathcal{O}(n_{dof}^2)$ for a standard matrix vector product.

3.1.3 FMM for the Maxwell's and the Helmholtz equations

Since the Green's function is identical, the same kind of algorithm can be used for the Helmholtz and the Maxwell equations. The initialisation and integration steps have to be specifically written, but the transfer step remains the same. These expressions are obtained by inserting the addition theorem (12) in the terms to compute, that is the left hand sides of either (1) or (7).

FMM for the Helmholtz equations

- **Initialization:** we compute the function $\mathcal{F}_\mathcal{C}$ defined on the unit sphere $\mathcal{S}$
$$\mathcal{F}_\mathcal{C}(\vec{s}) = \int_{x\in S\cap\mathcal{C}} \left[\lambda(x) - \vec{s}.\vec{\nu}_x\phi(x)\right] e^{ik\vec{s}.\vec{xM}}dx. \tag{17}$$

- **Integration:** we finish this calculus by integrating both on $\mathcal{S}$ and on $\Gamma\cap\mathcal{C}'$. We have two integrations formulae, one for λ^t function (third line of (1)), one for ϕ^t functions (first two lines of (1) which are identical as far as the distant interactions are concerned).
$$\begin{cases} \dfrac{k^2}{16\pi^2}\displaystyle\int_{y\in S\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} \mathcal{G}_{\mathcal{C}'}(\vec{s})\vec{s}.\vec{\nu}_y e^{ik\vec{s}.\vec{M'y}}\phi^t(y)d\vec{s}dy, \\ \dfrac{k^2}{16\pi^2}\displaystyle\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} \mathcal{G}_{\mathcal{C}'}(\vec{s})e^{ik\vec{s}.\vec{M'y}}\lambda^t(y)d\vec{s}dy. \end{cases}$$

The impedance η does not appear here because it is entirely treated as close interaction.

FMM for the Maxwell's equations

Since the waves propagate in the two media, one have to use two separate FMM algorithms, with different wave numbers k_e and k_i, different Green's kernels G_e and G_i. For each of these two computations, the initialisation and integration steps are defined as follows.

- **Initialization:** we compute the function $\vec{\mathcal{F}}_{\mathcal{C}}$ defined on the unit sphere $\mathcal{S}$

$$\vec{\mathcal{F}}_{\mathcal{C}}(\vec{s}) = \int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}} \left[\vec{j}(x) + \frac{1}{c\mu}\vec{s}\wedge\vec{m}(x)\right] dx.$$

  The component of $\vec{\mathcal{F}}_{\mathcal{C}}(\vec{s})$ tangential to $\vec{s}$ has then to be nullified [9].

- **Integration:** we finish this calculus by integrating both on $\mathcal{S}$ and on $\Gamma\cap\mathcal{C}'$. We have two integrations formulae, one for the $\vec{j}^t$ functions, one for the $\vec{m}^t$ functions.

$$\begin{cases} \dfrac{ik}{16\pi^2}\displaystyle\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.\vec{M'}y}\left(-\frac{i\omega\mu}{Z_0}\right)\vec{\mathcal{G}}_{\mathcal{C}'}(\vec{s}).\vec{j}^t(y)d\vec{s}dy, \\ \dfrac{ik}{16\pi^2}\displaystyle\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.\vec{M'}y}\left(\frac{-ik}{Z_0}\right)\vec{\mathcal{G}}_{\mathcal{C}'}(\vec{s}).(\vec{m}^t(y)\wedge\vec{s})d\vec{s}dy. \end{cases}$$

In the case of absorbing material (with a complex wave number), some precautions have to be taken in the numerical computations (see [3] for the details) but the global algorithm remains unchanged. Initialization and integration formulae for EFIE, MFIE and CFIE can be easily derived from those given above.

3.1.4 Advanced FMM

We have given an overview of the fast multipole method in its simplest version. In this section, we introduce some improvements to the algorithm or its implementation that we have developped in our work.

Multi-level algorithm

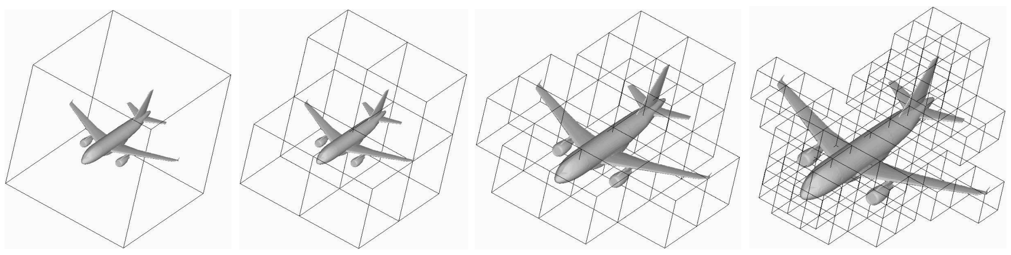


Figure 5: Subdivision of an aircraft through an octree

The multi-level fast multipole algorithm uses a recursive subdivision of the diffracting object as shown in Figure 5. The resulting tree is called an *octree*. The idea is to explore this tree from the root (largest box) to the leaves, and at each level to use the single level multipole method to treat interactions between non-neighbouring domains that have not yet been treated at an upper level. At the finest level, we treat classically the interactions between neighbouring domains. The improvement adds two new steps to the algorithm (ascent and descent), but the formulae given ealier for the Maxwell and the Helmholtz FMM remain unchanged. The optimal complexity of the multi level FMM is $\mathcal{O}(n_{dof}\log(n_{dof}))$ where n_{dof} is the number of degrees of freedom. All the results given in this paper use the multi-level FMM. We refer to iSection 4.1.1 where the benefit of using the FMM is illustrated.

Parallel algorithm

In order to widely exploit the available parallel architectures, we have developped a parallel distributed memory implementation of the FMM algorithm based on the message passing paradigm and the MPI library. The distribution of the octree is made level by level, each of them is equally distributed among the processors. When a given cell $\mathcal{C}$ is affected to a processor, all the computations required to build the radiation functions $\mathcal{F}_{\mathcal{C}}$ and $\mathcal{G}_{\mathcal{C}}$ are performed exclusively on this processor. Each phase of the calculation (initialisations, ascents or descents between two levels, transfers at a given level, integrations) is preceded by a communication phase. Therefore a complete multipole calculation on a parallel machine consists of an alternance of computation steps and communication steps. This requires a good load balancing at each level of the octree. We refer to 4.1.2 for an illustration of the parallel performance of this implementation.

Out-of-core algorithm

Since the aim of this method is to handle very large computations, we have also included an out-of-core feature: when the octree is too large to fit in memory, we divide each level of this tree in groups, and keep at most two of them in RAM. The rest goes to disk. The larger the number of groups is, the smaller the memory requirement (and the slower the code, due to disk I/O). Nevertheless, with the large amount of memory now available on computers, this feature is usefull only for extremely large number of unknowns (several millions per processor). We refer to 4.1.5 for the performance of this functionnality.

Adjustable accuracy

An interesting feature of the FMM is that it can be tuned to be more or less accurate and fast. By adjusting parameters such as the number of discretization points on the unit sphere, the size of the octree's leaves, the number of integration points on Γ, we can choose between being very accurate (and not so fast) or less accurate (and really fast). We have implemented in our FMM code three accuracy levels in order to exploit this feature in iterative solvers :

- Accurate FMM: the FMM is as accurate as possible (due to the divergence of the transfer function (13) when $L \to \infty$, the accuracy of the method is limited).
- Fast FMM: the FMM is as fast as possible while keeping the error criterium below a few percents.
- Intermediate FMM: this is the default level, located between the two others.

We refer to 4.1.4 for an illustration of this feature.

3.2 The parallel iterative linear solvers

3.2.1 The iterative solvers

We focuss now on the solution of the linear system

$$Ax = b$$

associated with the discretization of the wave propagation problem under consideration. As mentioned in the introduction, direct dense methods based on Gaussian elimination quickly becomes unpractical when the size of the problem increases. Iterative Krylov methods are a promising alternative in particular if we have fast matrix-vector multiplications and robust preconditioners. For the solution of large linear systems arising in wave propagation, we found that GMRES [20]

Algorithm 1 *:* Right preconditioned full-GMRES with MGS orthogonalization scheme

```
1:  Choose a convergence threshold $\varepsilon$ ;
2:  Choose an initial guess $x_0$ ;
3:  $r_0 = b - Ax_0$ ; $\beta = \|r_0\|$ ;
4:  $v_1 = r_0/\beta$ ;
5:  for $k = 1, 2, \dots$ do
6:      $z_k = Mv_k$ ;
7:      $w = Az_k$ ;
8:      for $i = 1$ to $k$ do
9:          $h_{i,k} = v_i^H w$ ;
10:         $w = w - v_i h_{i,k}$ ;
11:     end for
12:     $h_{k+1,k} = \|w\|$ ;
13:     $v_{k+1} = w/h_{k+1,k}$ ;
14:     Find $y$ of the least-squares solution of $\min_y \|e_1\beta - \bar{H}_k y\|$ where $e_1$ is the
        first column of the identity matrix of order $k+1$ and $\bar{H}_k$ the $(k+1)$-by-$k$
        Hessenberg matrix with nonzero entries defined by the $h_{i,j}$ ;
15:     if $(\|e_1\beta - \bar{H}_k y\| \le \varepsilon\|b\|)$ then
16:         Exit if $\|b - A(x_0 + M(V_k y))\| \le \varepsilon\|b\|$ ;
17:     end if
18: end for
19: Set $x = x_0 + M(V_k y)$
```

was fairly efficient [6, 27]. The GMRES algorithm with right preconditioner is described in Algorithm 1 where M denotes the preconditioner. Our approach to construct M is described later in Section 3.3.

To improve the robustness of GMRES on large problems we proposed [6] an embedded iterative scheme that combines Flexible GMRES (FGMRES) [18] and GMRES solvers with different FMM. The basic idea is to carry out a few steps of preconditioned GMRES for the preconditioning operation. The overall algorithm results in an inner-outer scheme where the outer solver is FGMRES and the inner scheme is GMRES. The default inner scheme consists of m steps of full-GMRES. However, we have also set a stopping criterion of the inner scheme. We design our stopping criterion based on the constraint that we do not want to solve the inner linear systems more accurately than what remains to be achieved in the outer scheme. In other words, we try to detect in the inner scheme when convergence might have occurred in the outer scheme. That is, if $\|r_k^{FGMRES}\|$ is the 2-norm of the residual of FGMRES at the k$-th$ iteration then inner GMRES is stopped at iteration ℓ if its associated residual $\|r_\ell^{GMRES}\|$ is such that $\|r_\ell^{GMRES}\| \le \varepsilon_{inner}^{(k)}$ (note that we used the fact that $\|z_k\| = 1$), where $\varepsilon_{inner}^{(k)}$ is determined by

$$\varepsilon_{inner}^{(k)} = \frac{\varepsilon_{outer}\|b\|}{2\|r_k^{FGMRES}\|}. \tag{18}$$

This stopping criterion enables the inner scheme to adapt to the outer scheme. The accuracy requested for the inner scheme is relaxed (set larger) while the convergence of the outer scheme occurs. We note that this relaxation strategy is closely related to [2].

The efficiency of this inner-outer scheme relies on two main factors: first, the inner solver has to be preconditioned so that the residual in the inner iterations can be significantly reduced in a few steps; then, the matrix-vector products within the inner and the outer solvers are carried

Algorithm 2 *:* full-Flexible GMRES with MGS orthogonalization

```
1: Choose a convergence threshold $\varepsilon$ ;
2: Choose an initial guess $x_0$ ;
3: $r_0 = b - Ax_0$ ; $\beta = \|r_0\|$ ;
4: $v_1 = r_0/\beta$ ;
5: for $k = 1, 2, \ldots$ do
6:   Perform $m$ step GMRES to solve $Az_k = v_k$;
7:   $w = Az_k$ ;
8:   for $i = 1$ to $k$ do
9:     $h_{i,k} = v_i^H w$ ;
10:    $w = w - v_i h_{i,k}$ ;
11:  end for
12:  $h_{k+1,k} = \|w\|$ ;
13:  $v_{k+1} = w/h_{k+1,k}$ ;
14:  Find $y$ of the least-squares solution of $\min_y \|e_1\beta - \bar{H}_k y\|$ where $e_1$ is the
     first column of the identity matrix of order $k+1$ and $\bar{H}_k$ the $(k+1)$-by-$k$
     Hessenberg matrix with nonzero entries defined by the $h_{i,j}$ ;
15:  if $\left(\|e_1\beta - \bar{H}_k y\| \leq \varepsilon\|b\|\right)$ then
16:    Exit if $\|b - A(x_0 + Z_k y)\| \leq \varepsilon\|b\|$ ;
17:  end if
18: end for
19: Set $x = x_0 + Z_k y$
```

out using different accuracies. In our scheme, an accurate FMM is used within the outer solver because the level of accuracy of this FMM governs the accuracy of the computed solution. A fast FMM is used within the inner solver. Since it is a preconditioner for the outer scheme, the inner iterations are just attempting to give a rough approximation of the solution and consequently do not require an accurate matrix-vector calculation.

It often happens that monostatic radar cross sections have to be computed. The procedure consists in considering a set of waves with the same wavelength but different incident angles that illuminate the object. For each of these waves we compute the field backscattered in the direction of the incident wave. This requires the solution of one linear system per incident wave. For a complete radar cross section calculation, from a few tens up to a few hundred waves have to be considered. We have then to solve a sequence of linear systems having the same coefficient matrix but different right-hand sides. The problem can be written

$$A(x_1, \ldots, x_p) = (b_1, \ldots, b_p).$$

If we were using a direct dense solver, designing an efficient algorithm would have been trivial. In the context of iterative methods, designing a good iterative solver with multiple right-hand sides is more challenging.

A natural question to address when solving a linear system with p right–hand sides is whether these right–hand sides are linearly independent or not. If the right–hand sides are linearly dependent with $\text{rank}([b_1, \ldots, b_p]) = q < p$, there exists U, an n_{dof}–by–q matrix, and S, a q–by–p matrix such that

$$B = US,$$

where $B = [b_1, \ldots, b_p]$. In such a case, a natural approach consists in solving the q systems associated with the right–hand sides U, that is

$$AX_U = U, \tag{19}$$

then recovering the unknowns of interest via

$$X = X_U S.$$

In the context of RCS, we have shown [13] that the right-hand sides were always strongly linear dependent and thus this strategy enables to considerably reduce the number of right-hand sides. We omit the details of this calculation and refer the reader to [13] where the detail on how obtaining U is described. The new set of linear systems (19) is then solved using a block variant of the flexible inner-outer scheme described above. The outer block solver is a block-GCR [26] and the inner solver a block-GMRES [19].

Most of the linear algebra kernels involved in the algorithms of this section (sum of vectors, dot products calculation) are straightforward to implement in a parallel distributed memory environment. The only two kernels that require to pay attention are the matrix-vector product and the preconditioning. The matrix vector product is performed with the parallel FMM implementation described in Section 3.1. The preconditioner is described in the next section, its design was constrained by the objectives that its construction and its application must be easily parallelizable.

3.3 The preconditioner

The design of robust preconditioners for boundary integral equations can be challenging. Simple parallel 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 [23]. Incomplete factorizations have been successfully used on nonsymmetric dense systems [21] and hybrid integral formulations [14], but on the EFIE the triangular factors computed by the factorization are often very ill-conditioned due to the indefiniteness of A. This makes the triangular solves highly unstable and the preconditioner uneffective.

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, 4, 5, 7, 29]). Owing to the rapid decay of the discrete Green's function, the location of the large entries in the inverse matrix exhibit some structure [1]. In addition, only a very small number of its entries have relatively large magnitude. This means that a very sparse matrix is likely to retain the most relevant contributions of the exact inverse. This remarkable property can be effectively exploited in the design of a robust approximate inverse.

The original idea of an approximate inverse preconditioner based on Frobenius-norm minimization is to compute the sparse approximate inverse as the matrix M which minimizes $\|I - AM\|_F$ subject to certain sparsity constraints. The Frobenius norm is chosen since it allows the decoupling of the constrained minimization problems into n_{dof} independent linear least-squares problems, one for each column of M, when preconditioning from the right. The independence of these least-squares problems follows immediately from the identity:

$$\|I - AM\|_F^2 = \sum_{j=1}^{n_{dof}} \|e_j - Am_{\bullet j}\|_2^2, \tag{20}$$

where e_j is the jth canonical unit vector, $m_{\bullet j}$ is the column vector representing the jth column of M and n_{dof} the dimension of the square matrices. Clearly, there is considerable scope for parallelism in this approach. The main issue is the selection of the sparsity pattern of M, that is the set of indices

$$S \;= \{\; (i,j) \;\subseteq\; [1, n_{dof}]^2 \; s.t. \; m_{ij} \neq 0 \;\}. \tag{21}$$

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. On boundary integral equations the discrete Green's function decays rapidly far from the diagonal, and the inverse of A may have a very similar structure to that of A [1]. The discrete Green's function can be considered as a column of the exact inverse defined 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, 5, 12]. When fast methods are used for the matrix-vector products, all the entries of A are not available and the pattern can be formed by exploiting the near-field part of the matrix that is explicitly computed and available in the FMM. In that context, relevant information for the construction of the pattern of M can be extracted from the octree.

In order to preserve sparsity, $\mathcal{O}(1)$ nonzeros locations are computed in the pattern of the column $m_{\bullet j}$ of M. Then each least-squares solution using the QR decomposition costs $\mathcal{O}(n_{dof})$, and the overall construction of M approximately $\mathcal{O}(n_{dof}^2)$ 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 strong local 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 S of M is known in advance, the nonzero structure for the jth column of M is automatically determined and defined as

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

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, and do not affect the solution of the least-squares problems (20). 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

$$min\|\hat{e}_j - \hat{A}\hat{m}_j\|_2, j = 1, ..., n_{dof}. \tag{22}$$

Usually problems (22) have much smaller size than problems (20) and can be effectively solved by a 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. Selecting more entries in $\tilde{A}$ than in M enhance the robustness of the preconditioner. Because of the complexity of the QR factorization, the computational cost to build the preconditioner is proportional to the number of nonzero entries in $\tilde{A}$ but quadratic with respect to the number of nonzeros in M. Experiments in [5] show the benefit of considering more entries in $\tilde{A}$ than in M.

The box-wise decomposition of the domain available in the FMM naturally leads to an *a priori* pattern selection strategy for M and $\tilde{A}$. 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 in one level of neighbouring boxes, and 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 problems corresponding to edges within the same 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 is, the larger the geometric neighbourhood that determines the sparsity structure of the columns of the preconditioner is. In the rest of the paper this preconditioner is denoted by M_{Frob}. 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.

4 Numerical experiments

In this section, we study the numerical behaviour of the code on the following geometries:

- a cetaf (see Figure 6(a)), a typical test case in electromagnetic simulations,
- an industrial civil aircraft form an European company (see Figure 6(b)), a real life model problem,
- a fake furtive aircraft (see Figure 6(c)),
- another civil aircraft (see Figure 6(d)),
- some spheres, geometries that are easy to mesh and to solve. Furthermore the analytic solution is known in some situations, which allows us to fully assess the quality of the solution computed by the code.

These objects have been tested with 4 equations :

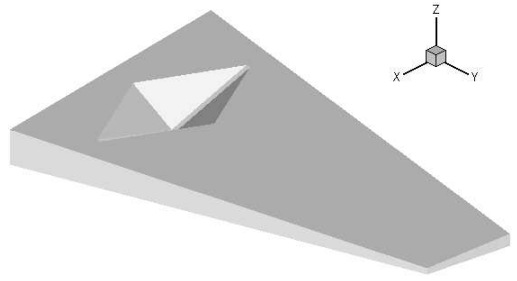
- Acoustic Imp : Acoustic case with impedance $\eta = 5$. The number of unknowns is the number of triangles + the number of vertices.
- Acoustic Rigid. The number of unknowns is the number of triangles.
- Electromagnetic Perfect Conductor : The object is placed in empty space. The number of unknowns is the number of edges.
- Electromagnetic Dielectric : The coefficients of the object are $\varepsilon_i = 13 + 3i$, $\mu_i = 1$. It is still placed in empty space. The number of unknowns is twice the number of edges.

4.1 Computational performance

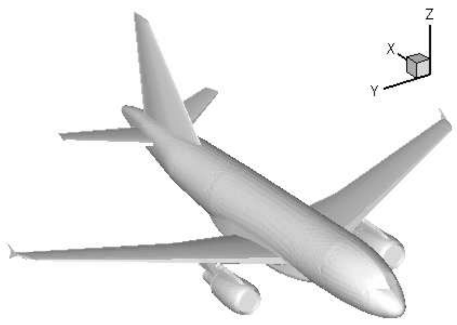
This section illustrate the raw performance of the multipole algorithm. More results and informations on the implementation tested here can be found in [27].

4.1.1 Sequential example

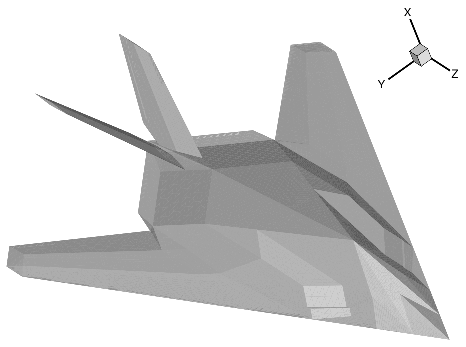
We compute the surfacic currents on a perfectly conducting sphere illuminated by a plane wave. We observe the accuracy of the method and its computational performance. We have run a set of tests on 32 spheres of radius 1 meter. The diameter increases from 1 to 32 wavelengths by step of 1 wavelength, whereas the number of unknowns n_{dof} grows from 972 to 1,16 million. The time needed for one FMM matrix-vector product (on a 866 MHz PC) is depicted as a function of n_{dof} in Figure 7, on the left. One can see that the growth is $\mathcal{O}(n_{dof})$ here. On the right is displayed the FMM accuracy. It is constant when the number of unknowns increases: there is no loss of accuracy when the number of levels in the octree grows, which means that our parameters in the FMM algorithm are well chosen. The memory requirements are approximately 0.5 kb of RAM and 4.6 kb of disk per unknown. We see that against all expectancy, everything here grows like $\mathcal{O}(n_{dof})$ instead of the well-known $\mathcal{O}(n_{dof} \log n_{dof})$. In fact, this scalability in $n \log n$ is asymptotic, and it is indeed observed for larger computations.



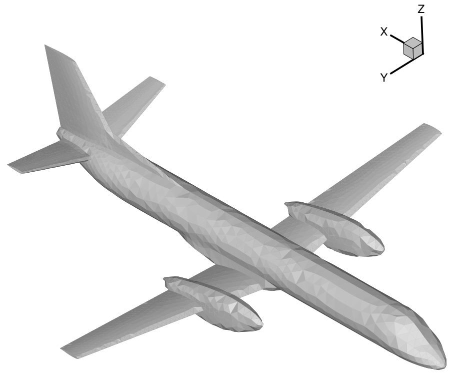
(a) Cetaf



(b) Aircraft 1

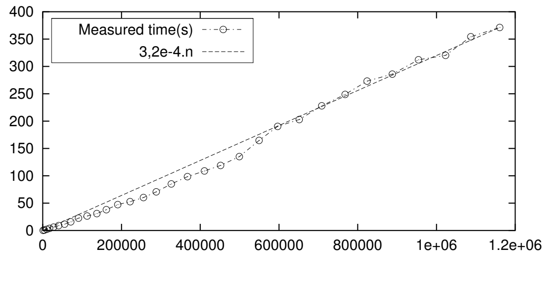


(c) Aircraft 2



(d) Aircraft 3

Figure 6: Mesh associated with test examples



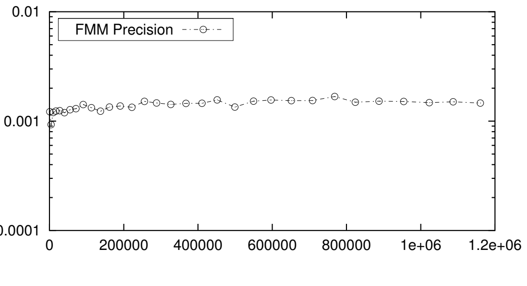


Figure 7: FMM product time (left) and accuracy (right)

4.1.2 Parallel scalability

In order to establish the performance of our parallel implementation, we now look at the CPU time required by one FMM product on a growing number of processors. For this purpose we select a perfectly conducting cetaf with 4,851,900 degrees of freedom (the choice of a sphere would have given much better but unrealistic results), running on 1 to 80 processors of IBM SP3. For each number of processors, we give the CPU time (in seconds) for one product, the parallel efficiency for this part of the code, and the percentage of this time spent in calculation (CPU), communication (COMM) and disk access (I/O).

Proc	Matrix-vector product				
	time	efficiency	% CPU	% COMM	% I/O
Out-of-core					
1	2884.1 s	-	67	-	33
4	728.8 s	99 %	72	17	11
In-core					
8	305.3 s	118 %	73	23	4
16	204.1 s	88 %	65	27	8
32	116.4 s	77 %	59	33	8
48	108.4 s	55 %	58	33	9
64	92.0 s	49 %	55	36	9
80	82.9 s	43 %	50	40	10

Table 1: FMM Parallel scalability on SP3

In this computation, the octree's size is 2.3 Gb. Therefore, the code runs in out-of-core mode on 1 and 4 processors, and it runs in-core afterwards. That is why we see an efficiency higher than 100 % for 8 processors. Up to 32 processors, the results are satisfactory, with an efficiency above 75 %. With 48 processors and above, the efficiency is not as good, but we must say that this test case is probably too small for such a large number of processors. We will see in the next section that a very high efficiency can also be achieved on 64 processors with larger problems.

4.1.3 Very large computation in parallel

Once again, we use a perfectly conducting sphere. Here, the diameter is 150λ, the number of degrees of freedom is 25.6 millions, the number of vertices is 8.5 millions. We solve the CFIE Equation (10) with a GMRES solver. The complete calculation requires 18 hours on a 64 processors of IBM-SP3, including 45 minutes for calculating the bistatic RCS in 1800 directions. 81 iterations are needed to reach a final normwise backward error ($\frac{\|Ax_m - b\|}{\|b\|}$ where x_m denotes the final iterate) below 10^{-5} Each iteration requires only 440 seconds, 13 % of this time is due to parallelism overhead, therefore the parallel efficiency is above 85 %. Figure 8 compares the RCS obtained with the exact result coming from the Mie series (entire curve on the left, closer view for the angles between 165 and 180 on the right). We observe a very good agreement. With more than 25 million unknowns, this is (as far as we know) the largest computation of this kind ever reported.

4.1.4 Adjustable accuracy level

Accuracy	Sphere 255,792		Cetaf 539,100	
Level	error(%)	time(s)	error(%)	time(s)
fast FMM	0.82	7.8	1.4	21.1
intermediate FMM	0.08	25.0	0.56	78.9
accurate FMM	0.04	28.0	0.39	102.9

Table 2: The adjustable accuracy level

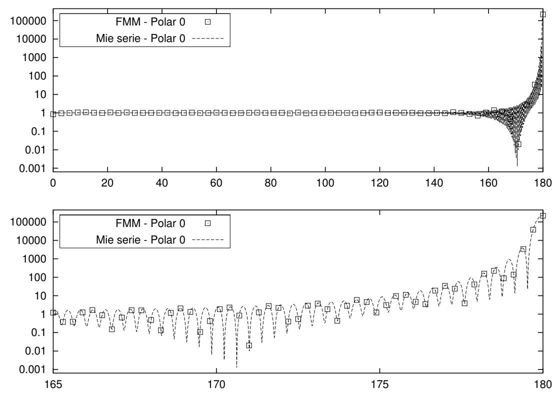


Figure 8: Bistatic RCS for VV polarization

Table 2 gives the relative error and the matrix-vector product time (on a 1.8 GHz Opteron workstation) for a perfectly conducting sphere with 255,792 unknowns, and a perfectly conducting cetaf with 539,100 unknowns. As expected, each accuracy level is slower than the previous one, whereas the accuracy increases noticeably. The differences in relative error between objects comes mainly from the quality of the mesh.

4.1.5 Out-of-core performance

Sphere 255,792				Cetaf 2,156,400			
Memory limit	RAM (Mb)	Disk (Gb)	Time (s)	Memory limit	RAM (Mb)	Disk (Gb)	Time (s)
∞	213.3	1.0	34.4	∞	1532.7	8.5	376.2
100	117.5	1.2	36.0	1000	1010.5	10.1	451.9
50	86.6	1.3	37.5	500	794.0	10.2	462.7

Table 3: The out-of-core feature performance on an Opteron workstation

Table 3 gives an illustration of the efficiency of the out-of-core implementation. The user gives the memory limit (first column) allowed for the radiation functions only. Since there are a few other pieces of data allocated, the RAM actually used (second column) is a bit above this limit. Nevertheless, these figures show that one can drastically reduce the memory consumption at the cost of a very low increase of the disk usage (third column) and the matrix-vector product time (fourth column).

4.1.6 M_{Frob} Parallel scalability

We now illustrate the parallel behaviour of the code observed on a HP-Compaq cluster of SMP. Each node consists of four Dec Alpha processors (EV 6, 1.3 GFlops peak) that share 4 GB of memory. All the runs have been performed in single precision. In Table 4, we demonstrate the parallel scalability of the implementation of the main components of the numerical software. We solve problems of increasing size on a larger number of processors, keeping the number of unknowns per processor constant. It can be seen that the construction of the preconditioner scales perfectly. This is a positive counterpart of the strategy that consists in keeping the size of the leaf-box constant. The application of this strategy requires some communication but still scales reasonably well. The scalability of the matrix-vector product is also satisfactory as the increase of the elapsed time is not only due to the amount of data exchanges but also to the $log(n)$ effect of its computation complexity.

n_{dof}	Nb procs	Construction time of M_{Frob} (sec)	Elapsed time precond (sec)	Elapsed time mat-vec (sec)
112908	8	513	0.39	1.77
221952	16	497	0.43	2.15
451632	32	509	0.48	2.80
900912	64	514	0.60	3.80

Table 4: Tests on the parallel scalability of the code with respect to the construction and application of the preconditioner and to the matrix-vector product operation on problems of increasing size. The test example is Aircraft 1

4.2 Numerical performance

In this section, we illustrate the efficiency and the robustness of the complete solver. In all the numerical experiments, the surface of the object is always discretized using ten points per wavelength. We consider right preconditioned GMRES method and the threshold for the stopping criterion is set to 10^{-3} on the normwise backward error $\frac{||r_m||}{||b||}$, where r_m denotes the residual at step m and b the right-hand side of the linear system. This tolerance is accurate enough for most of the engineering purposes. It enables to properly compute either a radar cross section or a surfacic field of the object. The initial guess is the zero vector and the target computer is again the HP-Compaq cluster. For the solution of the Maxwell's equations we only consider the EFIE formulation (8) that does not require any restriction on the geometry of the scattering obstacle. From a linear algebra point of view we mention that EFIE usually gives rise to linear systems that are more much difficult to solve with iterative methods that those associated with CFIE for instance.

We first assess the choice of the unsymmetric Krylov methods we have used. The matrix associated with the EFIE is complex symmetric in exact arithmetic and symmetric QMR [11] can be considered as an alternative method to GMRES. This method exploits the symmetry of the matrix in order to obtain a three term recurrence. This means that SQMR performs a constant number of dot products and axpy operations per iterations (as opposed to GMRES where at step k, $\mathcal{O}(k)$ dot products and axpy have to be performed for the orthogonalisation process). The associated drawback is an observed delay in the convergence due to the fact that the vectors are not anymore orthogonal. As shown in [11], the three term recurrence is preserved when preconditioning is used, provided that the preconditioner M is symmetric as well. Since the preconditioner computed via the Frobenius minimization approach is (in general) unsymmetric, symmetry can be restored by considering $M_{Frob}^{Sym} = \frac{M_{Frob}+M_{Frob}^T}{2}$. In our experiments, the matrix-vector products are performed using three different formulations of the FMM. Even though, in exact arithmetic, the dense matrix is symmetric; the use of the floating point arithmetic combined with the approximation made in the three implementations of the FMM deteriorate this property. We therefore end-up by using a non-symmetric matrix-vector product in a symmetric solver. In Figure 9 we plot the backward error $\frac{\|r_k\|}{\|b\|}$ as a function of the iterations for three formulations of the FMM (denoted by fmm1:fast, fmm2:medium and fmm3:accurate) and two different arithmetics (i.e. single or double precision). In Figure 9, we also plot the convergence of GMRES as a reference. The attainable accuracy of SQMR seems to correspond with the level of symmetry of the matrix; but this latter also greatly influences the rate of convergence. The better the symmetry is, the faster the convergence. When the matrix is nearly symmetric (double precision fmm1 or fmm3), the behaviour of SQMR is close to the one of GMRES. The SQMR method, which is very appropriate for those problems when the matrix is fully assembled [4], may also be applied with the multipole method but requires careful

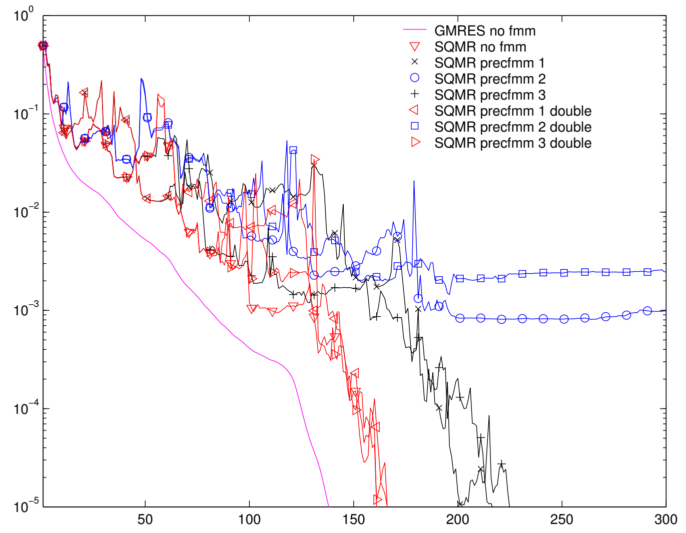


Figure 9: Convergence history of GMRES and SQMR with various fast multipole calculations

implementation to maintain the symmetry of the multipole expansion. At this point, experiments on bigger matrices are very disappointing and we have not been able to exploit the symmetry of the operator. In [4, 27], similar observations have been made for transpose free QMR (another short term recurrence solver). In [6], it is observed that the GMRES method was clearly outperformed by the embedded FGMRES/GMRES linear solver. In particular, this scheme succeeds to solve a 1 Mdof problem in EFIE formulation on the Aircraf 2 problem, while classic GMRES does not converge. For this reason, the FGMRES/GMRES scheme is the default solver used in the experiments of this paper.

In Table 5 to 8, we report on the numerical experiments performed on the four geometries depicted in Figure 6 for the solution of one linear system associated with the considered physical problem. For a given geometry the size of the mesh is the same for the four different calculations that are acoustic with impedance, acoustic with rigid boundary conditions, electromagnetic with perfect conductor and electromagnetic with dielectric. Nevertheless the number of degree of freedoms varies from one problem to the other depending on the nature of the problem. For each experiment, we give the number of degrees of freedom (n_{dof}), the maximum number of steps for the inner GMRES, the number of full-FGMRES iterations required to convergence and the associated accumulated number of GMRES steps performed in the inner scheme. This latter information is displayed in the form # FMRES + # GMRES, we remind that the FGMRES iterations are performed using an accurate FMM, while the inner GMRES steps implement a fast (and less accurate) FMM. In those tables, we also indicate the number of processors used for the simulation as well as the elapsed time required on the HP-Compaq.

From a linear algebra point of view, the linear systems associated with acoustic problem with rigid boundary conditions are the hardest to solve, but, are the smallest in size. The easiest ones are the linear systems coming from the acoustic problems with impedance or from the electromagnetic problems with dielectric. These latter problems are larger and require more computing resources. They are the most CPU consuming problems.

We mention that the use of the relaxation strategy was very beneficial for the acoustic problem with impedance. For instance on the Aircraft 3 problem, we have 3 outer iterations, the first performed 25 inner iterations, the second 16 and the last only 5. For that problem, the maximum of number of inner steps is set to 40 and the relaxation strategy always prevents to perform systematically the 40 inner steps.

n_{dof}	itermax-inner	# iter	# procs	Elapsed time
		Acoustic Imp.		
86,256	30	2 + 29	8	18mn
		Acoustic Rigid		
28,752	30	16+ 480	8	12mn
		Electromagnetic Perfect Conductor		
86,256	40	16 + 640	8	36mn
		Electromagnetic Dielectric		
172,512	40	6 + 240	16	54mn

Table 5: Numerical performance of the embedded schemes on the Cetaf problem

n_{dof}	itermax-inner	# iter	# procs	Elapsed time
		Acoustic Imp.		
94,706	30	3 + 42	8	42mn
		Acoustic Rigid		
31,570	30	45+ 1350	8	48mn
		Electromagnetic Perfect Conductor		
94,706	40	32 + 1280	16	48mn
		Electromagnetic Dielectric		
189,408	40	3 + 83	16	4h 36mn

Table 6: Numerical performance of the embedded schemes on the Aircraft 1 problem

n_{dof}	itermax-inner	# iter	# procs	Elapsed time
		Acoustic Imp.		
341,475	40	2+33	16	1h 54mn
		Acoustic Rigid		
113,827	40	17+680	16	1h 06mn
		Electromagnetic Perfect Conductor		
341,475	50	26 + 1300	40	1h 54mn

Table 7: Numerical performance of the embedded schemes on the Aircraft 2 problem

n_{dof}	itermax-inner	# iter	# procs	Elapsed time
		Acoustic Imp.		
341,475	40	3 + 48	16	2h 42mn
		Acoustic Rigid		
117,722	40	49+ 1960	16	4h 24mn
		Electromagnetic Perfect Conductor		
341,475	50	27 + 1350	16	5h 48mn

Table 8: Numerical performance of the embedded schemes on the Aircraft 3 problem

When monostatic radar cross sections (RCS) have to be computed a linear system with multiple right-hand sides has to be solved. Depending on the wavelength, from a few tens up to a few hundred waves have to be considered. As mentioned in Section 3.2.1 we first compute a basis of the space spanned by the set of right-hand sides. Then, we apply to the remaining right-hand sides a block extension of the FGMRES/GMRES scheme, the Block GCR/Block-GMRES solver, that could also be viewed as a block GMRESr [28]. This solver is able take advantage of the fact that all the right-hand sides are given simultaneously.

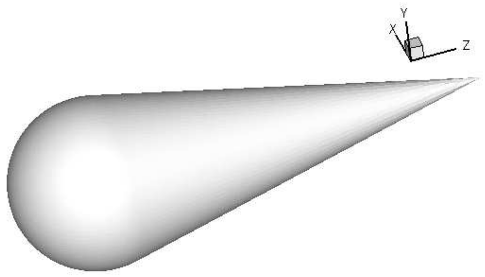


Figure 10: geometry of the coated-sphere used for the RCS calculation

To illustrate the benefit of the proposed scheme, we consider the coated-sphere test example, that is an electromagnetic problem with a thin dielectric layer (with relative electromagnetic coefficients $\varepsilon = 15+1.8i, \mu = 1$) placed on a perfectly conducting core (see Figure 10 for the geometry of this object). For that problem, the RCS calculation is built by considering 361 illuminating waves resulting in 361 right-hand sides. According to the targeted accuracy of the solution [13], the space spanned by these 361 vectors can be accurately enough represented in a space of dimension 30. This means that the solution of the 361 linear systems reduces to the solution of 30. The solution of those 30 linear systems using the block approach requires 90 matrix-vector products with an accurate FMM for the outer loops and 960 matrix-vector products with a fast FMM for the inner loop. The overall calculation requires 13h 36 mn of CPU time on the HP-Compaq. To assess the benefit of using this scheme, we mention that the solution of one of the 361 right-hand sides using the FGMRES/GMRES solver (i.e. our best linear solver when only one right-hand side has to be solved) requires 4 accurate FMM and 46 fast FMM calculations and consumed 7h 18 mn of CPU time on the same computer. This tremendeous saving introduced by the block-variant method is the consequence of two effects. The first effect is numeric. In Block GCR, the right-hand sides shared their Krylov spaces and therefore more information per right-hand side is available at each step. The second effect is the blocking of the matrix-vector product. Gathering the matrix-vector products is beneficial; it reduces considerably disk I/O, it enables a better parallelization of the algorithm and it better uses the memory hierarchy as the associated algorithm exhibits a good data locality.

5 Concluding remarks

In this paper we describe the building boxes of a parallel software that solves acoustic or electromagnetic problems in the frequency domain. The approach implemented in a parallel distributed memory environment exploits an efficient fast multipole technique that enables to treat huge problems in a reasonable amount of time using moderate parallel platforms. The largest calculation presented here has 25 Mdof. Its solution using a direct solver would require 254 TB of memory to store the matrix. On the largest and fastest parallel computer installed today, namely the Earth Simulator (41 Tflops, 10 TB memory), the matrix could not be loaded in core. Even if it could be stored in memory the factorization using a parallel direct solver would require 7 years of calculation assuming that the sustained computation is performed at peak performance. These latter figures illustrated the technological gap enabled by the FMM. For a more detailed presentation of this huge calculation we refer to [27]. To design a complete solver, the FMM that performs a matrix-vector product should be combined with efficient Krylov solvers and preconditioners. In that respect we describe an approximate inverse preconditioner that complies nicely with the constraints of parallelism and numerical efficiency. Furthermore it fits perfectly well in the framework of the FMM and can be naturally and efficiently implemented using the FMM data structures. Regarding the linear solvers, we illustrate the efficiency and the robustness of embedded Krylov schemes for solving large industrial problems arising from acoustic and electromagnetic applications. When only one right-hand side has to be solved the inner-outer scheme based on FGMRES and GMRES appears extremely effective. Its extension to deal with multiple right-hand sides, namely block GCR and block GMRES combined with an appropriated procedure to reveal the dimension of the space spanned by the right-hand sides also show its impressive efficiency. For details on that latter technique we refer to [13]. Implementing all these numerical techniques in a software package enables to develop a parallel simulation tool that can be used for research and engineering purposes to design new products requiring simulation of wave propagation phenomena on moderately parallel platforms.

6 Acknowledgments

We would like to thank G. Alléon for his stimulating support during the development of this work. We would also like to acknowledge the computing support provided by the CINES (Centre Informatique National de l'Enseignement Supérieur) and the CEA (Commissariat à l'Énergie Atomique) that enabled us to access their computing facilities.

References

[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] A. Bouras and V. Frayssé. Inexact matrix-vector products in Krylov methods for solving linear systems: a relaxation strategy. *SIAM J. Matrix Analysis and Applications*, 2004. To appear.

[3] Q. Carayol. *Développement et analyse d'une méthode multipôle multiniveaux pour l'électromagnétisme.* PhD thesis, Université Paris 6, janvier 2002.

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

[5] 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.

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

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

[8] R. Coifman, V. Rokhlin, and S. Wandzura. The Fast Multipole Method for the Wave Equation: A Pedestrian Prescription. *IEEE Antennas and Propagation Magazine*, 35(3):7–12, 1993.

[9] F. Collino and F. Millot. La méthode multipôle deux composantes pour l'électromagnétisme. Technical Report TR/EMC/01/22, CERFACS, 2001.

[10] E. Darve. *Méthodes multipôles rapides : Résolution des équations de Maxwell par formulations intégrales.* PhD thesis, Université Paris 6, juin 1999.

[11] R. W. Freund and N. M. Nachtigal. An implementation of the QMR method based on coupled two-term recurrences. *SIAM J. Scientific Computing*, 15(2):313–337, 1994.

[12] 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.

[13] J. Langou. *Iterative methods for solving linear systems with multiple right hand sides.* PhD thesis, Institut National des Sciences Appliquées de Toulouse, 2003.

[14] 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.

[15] J. C. Nédélec. Ondes acoustiques et électromagnétiques; équations intégrales. Technical report, Cours DEA, École Polytechnique, Paris, 1996.

[16] S. M. Rao, D. R. Wilton, and A. W. Glisson. Electromagnetic scattering by surfaces of arbitrary shape. *IEEE Transactions on Antennas and Propagations*, 30(3):409–418, may 1982.

[17] P.-A. Raviart and J. M. Thomas. A mixed finite element method for 2nd order elliptic problems. In *Mathematical aspects of finite element methods (Proc. Conf., Consiglio Naz. delle Ricerche (C.N.R.), Rome, 1975)*, pages 292–315. Lecture Notes in Math., Vol. 606. Springer, Berlin, 1977.

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

[19] Y. Saad. *Iterative Methods for Sparse Linear Systems.* SIAM, Philadelphia, 2003. Second edition.

[20] 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.

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

[22] X. Q. Sheng, J. M. Jin, J. Song W. C. Chew, and C. C. Lu. Solution of combined field integral equation using multilevel fast multipole algorithm for scattering by homogeneous bodies. *IEEE Ant. Propag*, 46(11):1718–1726, November 1998.

[23] J. Song, C-C Lu, and W. C. Chew. Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects. *IEEE Transactions on Antennas and Propagation*, 45(10):1488–1493, October 1997.

[24] J. M. Song and W. C. Chew. The fast illinois solver code : Requirements and scaling properties. *IEEE Computationnal Science and Engineering*, 5(3):19–23, July-September 1998.

[25] J. M. Song, C. C. Lu, W. C. Chew, and S. W. Lee. Fast illinois solver code. *IEEE Antennas and Propagation Magazine*, 40(3), June 1998.

[26] P. Soudais. Iterative solution of a 3–d scattering problem from arbitrary shaped multidielectric and multiconducting bodies. *IEEE Antennas and Propagation Magazine*, 42(7):954–959, 1994.

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

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

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