INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2003; **43**:865–879 (DOI: 10.1002/fld.501)

Performance of a parallel implementation of the FMM for electromagnetics applications

G. Sylvand*,†

CERMICS/INRIA, 2004 route des lucioles, BP 93, 06902 Sophia-Antipolis, Cedex, France

SUMMARY

This paper describes the parallel fast multipole method implemented in EADS integral equations code. We will focus on the electromagnetics applications such as CEM and RCS computation. We solve Maxwell equations in the frequency domain by a finite boundary-element method. The complex dense system of equations obtained cannot be solved using classical methods when the number of unknowns exceeds approximately 10^5. The use of iterative solvers (such as GMRES) and fast methods (such as the fast multipole method (FMM)) to speed up the matrix–vector product allows us to break this limit. We present the parallel out-of-core implementation of this method developed at CERMICS/INRIA and integrated in EADS industrial software. We were able to solve unprecedented industrial applications containing up to 25 million unknowns. Copyright © 2003 John Wiley & Sons, Ltd.

KEY WORDS: FMM; Maxwell; multipole; CEM; parallel

1. INTRODUCTION

We want to simulate the electromagnetic wave propagation phenomena in order to address various industrial problems (telecommunication antenna radiation patterns, electromagnetic compatibility, stealth design, etc.). We solve Maxwell equations in the frequency domain using an integral formulation and a finite-boundary-element method. For instance, if we consider a perfectly conducting object Ω illuminated by a monochromatic incoming plane wave E_{inc}, the magnetic current $\vec{j}(x)$ defined on $\Gamma = \partial\Omega$ satisfies $\forall \vec{j}^t$

$$\int_{\Gamma\times\Gamma} G(|y-x|)\left(\vec{j}(x)\cdot\vec{j}^t(y) - \frac{1}{k^2}\,\mathrm{div}_\Gamma\,\vec{j}(x)\,\mathrm{div}_\Gamma\,\vec{j}^t(y)\right)\mathrm{d}x\,\mathrm{d}y = \frac{i}{kZ_0}\int_\Gamma \vec{E}_{\text{inc}}(x)\vec{j}^t(x)\,\mathrm{d}x \qquad (1)$$

* Correspondence to: G. Sylvand, CERMICS/INRIA, 2004 route des lucioles, BP 93, 06902 Sophia-Antipolis, Cedex, France.
† E-mail: guillaume.sylvand@sophia.inria.fr; WebSite: http://www-sop.inria.fr/caiman/personnel/Guillaume.Sylvand/

Contract/grant sponsor: EADS Corporate Research Center, France

Received 3 March 2002
Revised 15 January 2003

Table I. Time and memory requirements growth for the three types of solvers.

	Direct solver	Iterative solver	Multipole solver
Assembly time	n_{dof^3}	n_{dof^2}	n_{dof}
Resolution time	$N_{\text{rhs}} n_{\text{dof}^2}$	$N_{\text{rhs}} N_{\text{iter}} n_{\text{dof}^2}$	$N_{\text{rhs}} N_{\text{iter}} n_{\text{dof}} \log n_{\text{dof}}$
Storage	n_{dof^2}	n_{dof^2}	$n_{\text{dof}} \log n_{\text{dof}}$

where $\vec{j}^t$ is a test current, $k = \omega/c$ is the wave number, and $G(|y-x|)$ is Green's function solution of $\Delta G + k^2 G = \delta_0$

$$G(|y-x|) = \frac{e^{ik|y-x|}}{4\pi|y-x|} \tag{2}$$

Equation (1) is usually referred to as electric field integral equation or EFIE. To discretize it, we use a boundary triangle mesh with Rao–Wilton–Glisson finite elements [1], where unknowns are carried by edges. In the case of perfectly conducting closed bodies, a magnetic field integral equation (MFIE) is also available. The linear combination of EFIE and MFIE is usually called combined field integral equation (CFIE), and is known to be the most efficient formulation in connection with iterative solvers. In the more general case of heterogeneous dielectric objects, the integral formulation is slightly more complicated, but the fast multipole method still applies in the same way.

This integral approach of Maxwell equations has several advantages over a classical volumic formulation. First, only the surface of the considered object Ω carries unknowns and there is no need to mesh the empty space around and inside it. Second, the radiation condition at infinity is directly included in the formulation. This method leads to the resolution of a full dense complex linear system, which can be solved either by a direct method (LU factorization for example) or an iterative method (GMRES, QMR, CG, etc.). When the number of unknowns n becomes large (more than $\approx 10^4$), these two methods (direct and iterative) become very expensive in CPU time and storage requirements as shown in Table I (where n_{dof} is the number of degrees of freedom, N_{rhs} is the number of right-hand sides and N_{iter} is the number of matrix–vector products required by the iterative solver for each right-hand side).

Unfortunately, in the electromagnetic computations we plan to do in the future, the number of unknowns will rather be counted in *millions* than in thousands. For instance, any computation on a full aircraft at 1 GHz will immediately ask for more than 1 million unknowns, which can hardly be solved even on the largest supercomputers available.

The fast multipole method, combined with any iterative solver, is an efficient way to overcome these limitations [2, 3]. The fast multipole method (FMM) is a new way to compute fast (but approximate) matrix–vector products with time and memory requirements in $n_{\text{dof}} \log n_{\text{dof}}$. In Table I, we call 'Multipole solver' an iterative solver using the fast multipole product instead of the usual one.

In order to see the actual benefits of this method, let us give a small example: we want to compute the radar cross-section of a wing-shape object called 'Cetaf' meshed with (only!) 5391 unknowns. Table II gives the total CPU time (in s) and disk space (in Mb) required by the three solvers. The RCS obtained with each solver is drawn in Figure 1: there is absolutely

Table II. Small example.

	Direct solver	Iterative solver	Multipole solver
CPU time	4871 s	1206 s	152 s
Storage	467 Mb	474 Mb	28 Mb

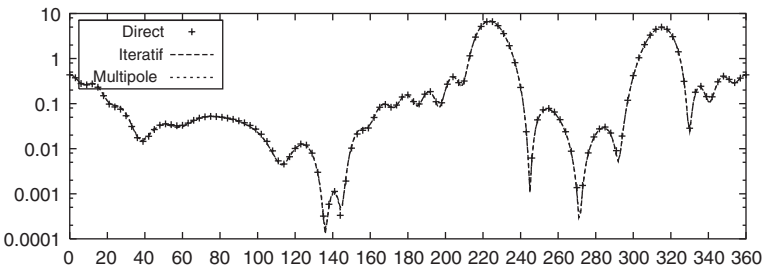

Figure 1. Cetaf mesh and radar cross-section.

no difference. We see that even on small problems, the FMM is able to give very accurate results at very low costs in time and storage.

2. THE FAST MULTIPOLE METHOD

As we said, the goal of the FMM [4] is to compute fast (but approximate) matrix–vector products. It is fast in the sense that CPU time is $\mathcal{O}(n\ln(n))$ instead of $\mathcal{O}(n^2)$, 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 usually below the error induced by the iterative solver or by the surfacic triangle approximation.

2.1. The one-level FMM

We will now give an overview of the FMM in its one-level version. For a matrix–vector product, the input data is a given surfacic current $\vec{j}(x)$ defined on Γ. We want to compute the left-hand side of the EFIE (1) for all test functions $\vec{j}^t$. First, we divide Γ into equally sized patches. A simple way to do that is to compute the intersection of Γ with a cubic grid (cf. Figure 2). The degrees of freedom are then dispatched between these cubic cells. Interaction of basis functions located in neighbour cells (that is to say cells that share at least one vertex) are treated as before.

The interactions of unknowns located in non-neighbour cells are accelerated with the FMM. The base of this algorithm if the following addition theorem: given two points x and y located in two distant (= non-neighbour) cells $\mathscr{C}$ and $\mathscr{C}'$ centred in M and M', we have

$$G(|y-x|) = \frac{ik}{16\pi^2} \lim_{L\to+\infty} \int_{\vec{s}\in\mathscr{S}} e^{ik\vec{s}\cdot x\vec{M}}\, T^L_{M\vec{M}'}(\vec{s})\, e^{ik\vec{s}\cdot M'y}\, d\vec{s} \tag{3}$$

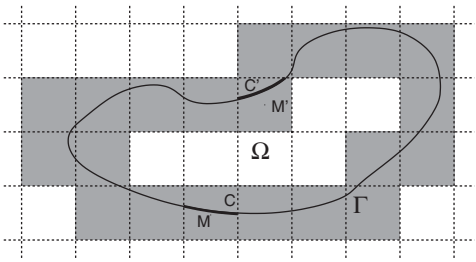

Figure 2. Division of Γ into patches (2D version).

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

$$T^L_{M\vec{M}'}(\vec{s}) = \sum_{0 \leqslant l \leqslant L} (2l+1) i^l h_l^{(1)}(k \cdot |M\vec{M}'|) P_l(\cos(\vec{s}, M\vec{M}')) \tag{4}$$

The parameter L is called number of poles. It is chosen in accordance with the size of the cell edge a, in order to have a good accuracy in (3) and no divergence in (4): $L = \sqrt{3}ka$ satisfies these conditions. According to the EFIE (1), the action of the current $\vec{j}(x)$ for $x \in \Gamma \cap \mathscr{C}$ on the basis function $\vec{j}^t$ located in $\mathscr{C}'$ is given by

$$\int_{x \in \Gamma \cap \mathscr{C}} \int_{y \in \Gamma \cap \mathscr{C}'} G(|y-x|) \left(\vec{j}(x) \cdot \vec{j}^t(y) - \frac{1}{k^2} \operatorname{div}_\Gamma \vec{j}(x) \operatorname{div}_\Gamma \vec{j}^t(y) \right) \mathrm{d}x \, \mathrm{d}y \tag{5}$$

Inserting the addition theorem (3) in this formula, we notice the appearance of the three steps of the one-level FMM.

Initialization: We compute the function $\mathscr{F}_\mathscr{C}$ defined on the unit sphere $\mathscr{S}$

$$\mathscr{F}_\mathscr{C}(\vec{s}) = \int_{x \in \Gamma \cap \mathscr{C}} \mathrm{e}^{\mathrm{i}k\vec{s} \cdot x\vec{M}} \vec{j}(x) \, \mathrm{d}x \tag{6}$$

$\mathscr{F}_\mathscr{C}$ depends only on the current $\vec{j}$, on the cell $\mathscr{C}$ and its centre M. It represents the action of the current $\vec{j}(x)$ for $x \in \Gamma \cap \mathscr{C}$ on *any* distant cell $\mathscr{C}'$.

Transfer: We multiply $\mathscr{F}_\mathscr{C}$ by the transfer function $T^L_{M\vec{M}'}$. The result is still a function defined on the unit sphere, and it represents the action of the current $\vec{j}(x)$ for $x \in \Gamma \cap \mathscr{C}$ *specifically* on $\mathscr{C}'$.

Integration: We finish this calculus by integrating both on $\mathscr{S}$ and on $\Gamma \cap \mathscr{C}'$

$$\frac{ik}{16\pi^2} \int_{y \in \Gamma \cap \mathscr{C}'} \int_{\vec{s} \in \mathscr{S}} [T^L_{M\vec{M}'}(\vec{s}) \mathscr{F}_\mathscr{C}(\vec{s})] \cdot \mathrm{e}^{\mathrm{i}k\vec{s} \cdot M'y} \vec{j}^t(y) \, \mathrm{d}\vec{s} \, \mathrm{d}y \tag{7}$$

There are several ways to handle the 'div div' term in (5), the simplest one is to consider that the vectors such as $\vec{j}$ and $\vec{j}^t$ have four components: $(j_x, j_y, j_z, \operatorname{div}_\Gamma(\vec{j})/k)$. Therefore, $\mathscr{F}_\mathscr{C}$ also has four components. These number of components can be reduced to two [5].

We can say a few words about the integral equations for acoustic diffraction problems. In the case of a rigid body hit by an incoming plane wave u_{inc}, we denote by u the air pressure,

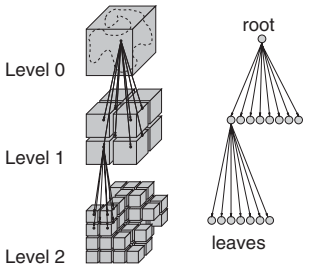

Figure 3. Octree.

and the unknown $\phi(x) = u^-_{|\Gamma}(x) - u^+_{|\Gamma}(x)$ is given by the following equation:

$$\oint_{\Gamma\times\Gamma} \frac{\partial^2 G}{\partial v_x \partial v_y} \phi(x)\phi^t(y)\,\mathrm{d}y\,\mathrm{d}x = \int_\Gamma \frac{\partial u_{\mathrm{inc}}(x)}{\partial v}\,\phi^t(x)\,\mathrm{d}x \tag{8}$$

From (3), we deduce the following decomposition for the derivative of G:

$$\frac{\partial^2 G}{\partial v_x \partial v_y} = \frac{ik}{16\pi^2} \lim_{L\to+\infty} \int_{\vec{s}\in\mathscr{S}} k\vec{s}\cdot\vec{v}_y \mathrm{e}^{\mathrm{i}k\vec{s}\cdot y\vec{M}} T^L_{M\vec{M}'}(\vec{s}) k\vec{s}\cdot\vec{v}_x \mathrm{e}^{\mathrm{i}k\vec{s}\cdot\vec{M'}x}\,\mathrm{d}\vec{s} \tag{9}$$

Hence, we can write a multipole algorithm with the following formula.
Initialization: $\mathscr{F}_{\mathscr{C}}$ is given by

$$\mathscr{F}_{\mathscr{C}}(\vec{s}) = \int_{y\in S\cap\mathscr{C}} k\vec{s}\cdot\vec{v}_y \mathrm{e}^{\mathrm{i}k\vec{s}\cdot y\vec{M}} \phi(y)\,\mathrm{d}y \tag{10}$$

Integration: We integrate the following term:

$$\frac{ik}{16\pi^2} \int_{x\in S\cap\mathscr{C}'} \int_{\vec{s}\in\mathscr{S}} \mathscr{G}_{\mathscr{C}}(\vec{s}) k\vec{s}\cdot\vec{v}_x \mathrm{e}^{\mathrm{i}k\vec{s}\cdot\vec{M'}x} \omega_i(x)\,\mathrm{d}\vec{s}\,\mathrm{d}x \tag{11}$$

In this case, the radiation functions have only one component.

2.2. The multi-level FMM

The multi-level FMM is an extension of the previous work. It is based on a recursive subdivision of Ω using an octree (cf. Figure 3). The effective result of this subdivision on a real object is shown in Figure 4.

In this case, the interaction between two degrees of freedom will be treated in this tree 'as high as possible', that is to say at the higher level where they still are in non-neighbour cells. If there exists no such level, the interaction is treated classically outside of the FMM. The multipole algorithm can now be written in four steps.

Step 1 (*Initialization*): For each leaf $\mathscr{C}$, we calculate $F_{\mathscr{C}}(\vec{s})$ which represents the influence of $\mathscr{C}$ on the outside of $\mathscr{C}$ in direction $\vec{s}\in\mathscr{S}^2$.

Step 2 (*Ascent*): We recursively calculate $F_{\mathscr{C}}$ at levels $n-1, n-2, \ldots, 3, 2$ (using $F_{\mathscr{C}}$ of the lower level).

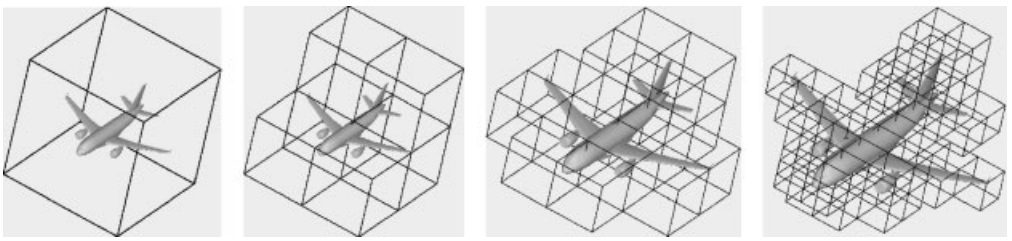

Figure 4. Subdivision of an Airbus A318 through an octree.

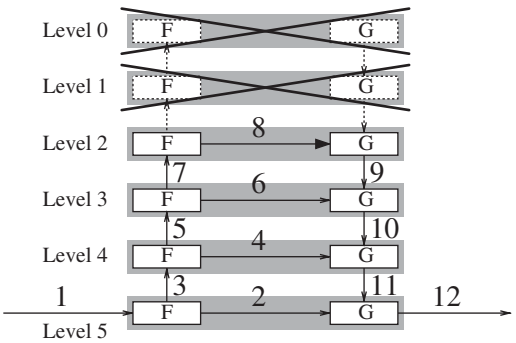

Figure 5. Multilevel multipole algorithm.

Step 3 (*Transfer and descent*): We recursively calculate $G_{\mathscr{C}}$ (which represents the influence of the outside of $\mathscr{C}$ on $\mathscr{C}$) at level 2, $3,\ldots,n-2,n-1$ (using $G_{\mathscr{C}}$ of the upper level, and $F_{\mathscr{C}}$ of the same level).

Step 4 (*Integration*): For each unknown $\vec{j}^t$ located in $\mathscr{C}$, using $G_{\mathscr{C}}$ we calculate the corresponding co-ordinate of the matrix–vector product.

This part of the algorithm is very technical, and has been extensively explained in References [4, 6]. Figure 5 gives a symbolic representation of how it works.

As we can see, levels 0 and 1 are not used; indeed, at these levels all cells are neighbour, and therefore all the interactions must be treated at lower levels. But it is not mandatory to ascend up to level 2. In fact, we can freely choose the highest level of our exploration of the octree. At this level, all the interactions not yet taken into account must be dealt with. Another important point is the fact that the number of poles L will increase with the size of the cells as we go up in the octree. One can show [4, 7], that the optimal complexity of this multilevel FMM is $\mathscr{O}(n\log(n))$ where n is the number of unknowns.

3. SEQUENTIAL PERFORMANCE OF THE FMM

3.1. Presentation of our implementation

The code that we have developed is written mainly in C, with a bit of fortran 77 (exclusively for numerical calculations). We have tried to offer as many parameters to adjust as possible.

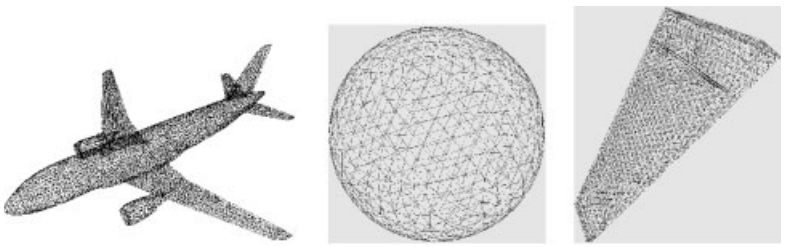

Figure 6. Three objects used for our tests: aircraft, sphere, cetaf.

Table III. Sequential performance of the FMM.

Shape	Aircraft	Sphere	Cetaf
Number of unknowns	213 084	1 023 168	1 056 636
Diameter in wavelengths	43λ	30λ	70λ
FMM levels	9	8	10
Residual	$5,6 \times 10^{-3}$	10^{-4}	10^{-2}
Number of iterations	100	37	43
Machines used	PC 866 MHz	PC 866 MHz	DEC 667 MHz
Memory(RAM)	159 Mb	460 Mb	500 Mb
Memory(Disk)	1.2 Gb	5.1 Gb	5.2 Gb
Total time	3.3 h	5.6 h	11.5 h

For instance, the user can choose the size of the leaves, the higher explored level of the octree, the FMM accuracy level, the number of integration points on Γ, the in-core or out-of-core feature, the 'simple' or 'double' precision in real numbers representation, etc. A long process of optimization has been led in order to find the optimal value for each of these parameters. We have also optimized the algorithm itself, especially the use of transfer functions (step 3), which have been sparsified, stored, reused and symmetrized. Through the use of high performance library such as BLAS, LAPACK and FFTW, we have been able to achieve very complex computation on a wide variety of architectures (PC, SGI, IBM, Sun, DEC, Cray). All details can be found in Reference [8]. We will now present a few results in order to demonstrate the interest of the FMM, and the performance of our implementation. Figure 6 presents the three kinds of objects that we have used during our tests: an aircraft, a sphere and a wing-shaped object called cetaf. These three objects are perfectly conducting and closed. The iterative solver used in GMRES [9] with 20 for restart parameter. In order to reduce the number of iterations, the CFIE formulation will be used.

3.2. Sequential FMM computations

In Table III, we have reported the main information concerning three sequential multipole computations. We can see that even on 'modest' workstations (such as PC), the FMM code is able to handle and solve very large linear systems. Without the FMM, a 1 million unknowns

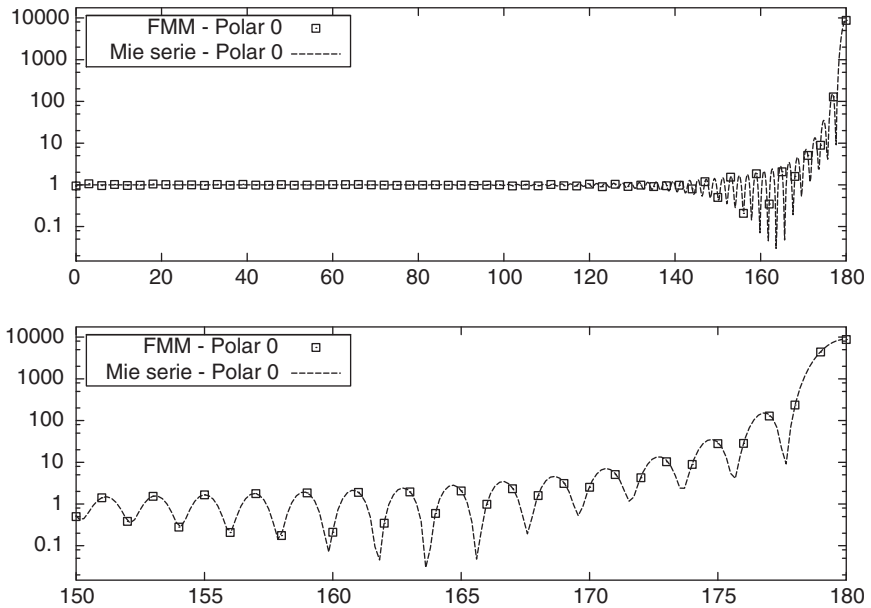

Figure 7. RCS of the 1 million unknown sphere (top: full curve, bottom: close-up).

Table IV. The out-of-core feature performance.

Number of groups	RAM Max. (Mb)	Time by iteration (s)
0(in-core)	1369	579.8
1	1070	590.6
2	649	636.9
3	545	658.6

case would require 18 terabytes of disk space, and several months of CPU time on the same machine. Figure 7 shows the radar cross-section in VV polarization of the 1 million unknowns sphere obtained through this computation and the exact solution, obtained with the Mie series. One can see the very good agreement between the two solutions.

3.3. The out-of-core feature

Since the aim of this method is to handle very large computations, we have 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, the smaller the memory need (and the slower the code, due to disk access).

Table IV gives an illustration of how well it works. We use a cetaf with 2 156 400 unknowns on one processor of SP3. Thanks to a home-made library, we know the exact maximum amount of memory used by the multipole code. When the number of groups increases

Table V. The adjustable accuracy level.

	Sphere 255 792		Cetaf 539 100		Aircraft 213 084	
Accuracy level	Error (%)	Time (s)	Error (%)	Time (s)	Error (%)	Time (s)
Fast FMM	0.94	42.6	1.6	157.3	2.0	68.7
Intermediate FMM	0.15	83.0	0.50	276.3	1.3	131.8
Accurate FMM	0.05	157.5	0.12	371.9	0.36	189.2

from 0 (which means in-core computation) to 3, the memory used is divided by 2.5 for a mere 13% time overcost for each matrix–vector product.

3.4. Adjustable accuracy level

By adjusting the FMM parameters such as the number of poles, the number of integration points, the sparsifying threshold for the transfer matrices, we can adjust the accuracy (and the speed) of the computation to a certain extent. This is a very interesting feature that can be used, for example, in flexible preconditioner or in FMM-specific solvers. We have therefore defined three accuracy levels:

Accurate FMM: The FMM is as accurate as possible (due to the divergence of the transfer function (4) when $L \to \infty$, the accuracy of the method is limited).

Fast FMM: The FMM is as fast as possible while keeping the error criterion below a few percent.

Intermediate FMM: This is the default level, located between the other two levels.

Table V gives the relative error and the matrix–vector product time (on a 866 MHz PC) for three objects with number of unknowns between 213 084 and 539 100. Each accuracy level is approximately two times slower than the previous one, whereas the accuracy increases noticeably. The differences in relative error between the three objects comes mainly from the size of the larger elements in each mesh.

3.5. Numerical scalability

If we count the number of operations needed to accomplish one matrix–vector product with the fast multipole algorithm, we have to make an assumption concerning the number of poles L: usually, in order to simplify the counting, it is taken proportional to the diameter of the boxes d at a given level, $L = kd$. With this formula, we find that the complexity of the algorithm is $\mathcal{O}(n_{\mathrm{dof}}^{3/2})$ for $n_{\mathrm{dof}} \gg 10^7$, and $\mathcal{O}(n_{\mathrm{dof}} \log n_{\mathrm{dof}})$ before. In practice, the number of poles is chosen according to the following formula:

$$L(d) = kd + C_\varepsilon \log_{10}(kd + \pi) \tag{12}$$

where the parameter C_ε is taken between 2 and 7, depending on the accuracy level. We have run a set of tests on 32 spheres with $C_\varepsilon = 7$. 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 drawn as a function of n_{dof} in Figure 8, on the left. One can see that the growth is $\mathcal{O}(n_{\mathrm{dof}})$ here. On the right is drawn the FMM accuracy. It is constant when the number of unknowns

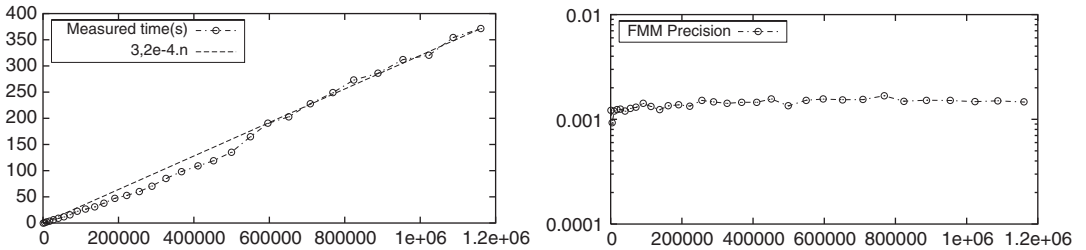

Figure 8. FMM product time (left) and accuracy (right).

Table VI. FMM numerical scalability.

Memory	
RAM	0.5 kb by unknown
Disk	4.6 kb by unknown
Time	
Assembly (s)	$3.7 \times 10^{-3} n_{\text{dof}}$
Iterations (s)	$3.2 \times 10^{-4} n_{\text{dof}}$

increases: there is no loss of accuracy when the number of levels in the octree grows, which means that our formula 12 is well chosen.

In Table VI, we show the time and memory requirements we have obtained during our tests. Against all expectancy, everything here grows like $\mathcal{O}(n_{\text{dof}})$ instead of the well-known $\mathcal{O}(n_{\text{dof}} \log n_{\text{dof}})$. In fact, when we count the number of operations, we use (12) with $C_\varepsilon = 0$. We underestimate the number of poles at every levels, *but the relative error on* $L(d)$ *that we introduce is larger for small values of d. Therefore, in our count we have underestimated the low levels, and overestimated the high levels*. The result $\mathcal{O}(n_{\text{dof}} \log n_{\text{dof}})$ comes from the fact that the estimated amount of computation at each level is $\mathcal{O}(n_{\text{dof}})$, and the number of levels is $\mathcal{O}(\log n_{\text{dof}})$. As we have seen, the actual amount of computation at each level is decreasing as we get closer to the root. Hence, we are not surprised to obtain an actual complexity lower than the estimation. Nevertheless, this does not mean that FMM is an $\mathcal{O}(n_{\text{dof}})$ algorithm, since we do not know how the code behaves when the number of unknowns n_{dof} increases even more.

4. PARALLEL IMPLEMENTATION AND PERFORMANCE

4.1. Overview of the parallelization

In order to have access to the wider choice of parallel architectures, we have chosen to use the message passing paradigm and the MPI implementation. Therefore, we have the possibility to use both clusters of PC (at INRIA for instance) or supercomputers with distributed or

Table VII. Example of multipole performance on supercomputers.

Shape	Cetaf	Sphere
Number of unknowns	2 156 400	11 360 748
Diameter in wavelengths	100λ	100λ
FMM levels	10	10
Number of processors	32	16
Formulation	EFIE+SPAI	CFIE
Residual	2.9×10^{-2}	10^{-4}
Number of iterations	100	49
Memory(RAM)	12 Gb	24.3 Gb
Memory(Disk)	63.8 Gb	132.8 Gb
CPU time	4.5 h	9 h

shared memory (IBM SP3 and SGI Origin 3800 at CINES[‡]). The distribution of the octree is made level by level, each of them is being fairly distributed among the processors. When a given cell $\mathscr{C}$ is affected to a processor, all the computations required to build the radiation functions $\mathscr{F}_{\mathscr{C}}$ and $\mathscr{G}_{\mathscr{C}}$ are made exclusively on this processor. Each phase of calculation (symbolized by 12 arrows in Figure 5) is preceded by a phase of communication. Therefore, a full multipole product on a parallel machine will consist of an alternance of computation steps and communication steps. This requires a good load balancing at each level of the octree.

4.2. *Performance on supercomputers*

In the first time, we show in Table VII the performance achieved on an SGI origin 3800. Both objects are perfectly conducting, illuminated by a plane wave. The solver is still GMRES, used in conjunction with a parallel SPAI preconditioner in the cetaf case. We can see that the time and memory requirements remain moderate considering the number of unknowns. In the next section, we will exhibit convincing results obtained on an even larger structure, that is why we do not show any graphical results proving the validity of these tests.

4.3. *Performance on clusters*

Clusters of PC have gained wide acceptance thanks to their increasing performance and low price. For instance, INRIA has built a cluster composed of 19 nodes (with two pentium III at 933 MHz, 512 Mb of RAM and 9 Gb of disk each). The nodes are connected through a fast-ethernet full-duplex switch. Table VIII shows some computations that can be done on this configuration. A 1 million unknowns industrial case has been completely treated in 5 h using only eight nodes (16 processors). Without the FMM, this kind of case would have required hundreds of processors and several days of computation on supercomputers hundreds of time more expensive.

[‡]http://www.cines.fr

Table VIII. Example of multipole performance on cluster.

Shape	Aircraft	Aircraft
Number of unknowns	213 084	1 160 124
Diameter in wavelengths	43λ	87λ
FMM levels	9	10
Number of processors	8	16
Solver	CFIE	CFIE
Residual	10^{-2}	10^{-2}
Number of iterations	81	91
Memory (RAM)	262 Mb	3.0 Gb
Memory (Disk)	2.1 Gb	16 Gb
CPU time	1.0 h	4.9 h

Table IX. FMM parallel scalability on SP3.

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

4.4. Parallel scalability

In order to establish the performance of our parallelization scheme, we will now take a look at the CPU time required by one FMM product on a growing number of processors. The case chosen is a 4 851 900 unknowns cetaf (the choice of a sphere would have given much better but unrealistic results), running on 1–80 processors of IBM SP3. For each number of processors, we give the CPU time (in s) 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). (Table IX).

In this computation, the octree's size is 2.3 Gb. Therefore, the code runs in out-of-core mode on one and four processors, and it runs in-core afterwards. That is why we see an efficiency higher than 100% for eight processors. Up to 32 processors, the results are good, 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 nodes. We will

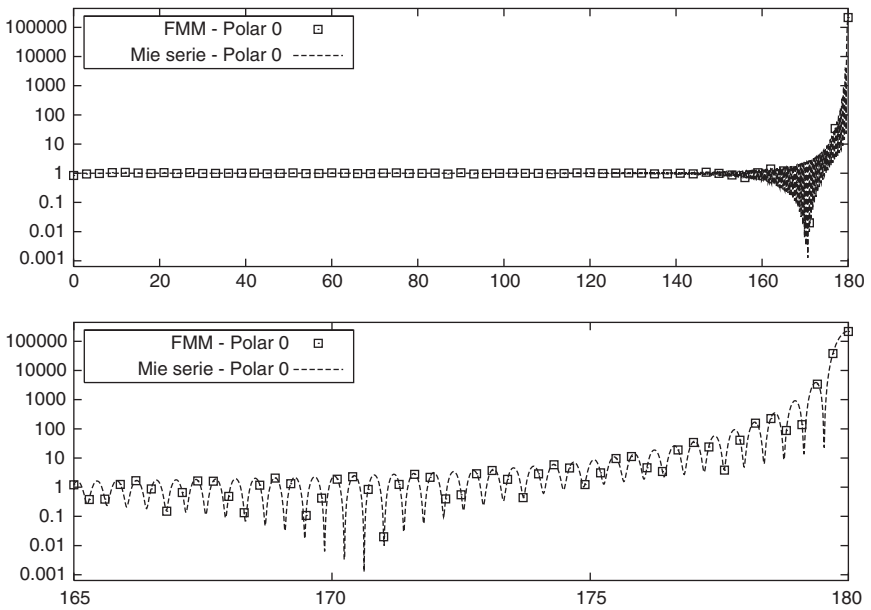

Figure 9. Bistatic RCS for VV polarization.

see in the next section that very high efficiency can also be achieved on 64 processors with larger cases.

5. EXTREMELY LARGE COMPUTATIONS

We will conclude this paper with two very large computations that demonstrate the full potential of our FMM implementation.

5.1. Large sphere RCS

Once again, we use a perfectly conducting sphere. Here, the diameter is 150λ and the number of degrees of freedom is 25.6 millions. We solve the CFIE with a GMRES solver. Eight-one iterations are needed to reach a final residual below 10^{-5}. Each iteration requires only 440 s, 13% of this time is used by parallelism; therefore, the parallel efficiency is above 85%. The whole calculation requires 18 h on a 64 processors SP3, including 45 min for calculating the bistatic RCS in 1800 directions. Figure 9 compares the RCS obtained with the exact result coming from the Mie series (entire curve on the top, closer view for the angles between 165 and 180 on the bottom). Once again, 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.

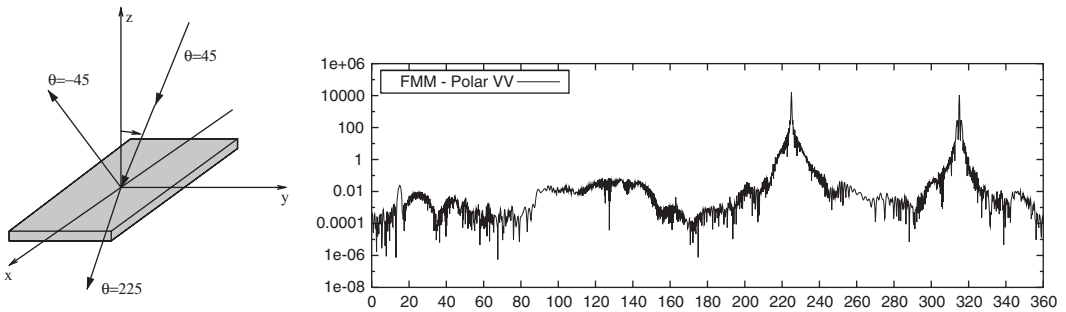

Figure 10. Cetaf case: definition and results.

5.2. *Industrial case*

We will now present a case in which we have used an SPAI preconditioner in order to improve the convergence. For that purpose, we have developed a parallel octree-based SPAI preconditioner. We will use it to compute the radar cross-section of a 13.5 million unknowns perfectly conducting cetaf (of diameter 250λ). On a 64 processor SP3, 12 h are needed the compute the preconditioner matrix. The preconditioned GMRES only needs 37 iterations to converge to a final residual below 10^{-2} (which is more than enough to obtain a correct RCS). Each multipole product requires 250 s, with only 42 s (17%) for the parallelism. Hence, the parallel efficiency is above 80%.

Figure 10 (left) presents the description of our testcase: an incoming plane wave comes from ($\theta = 45,\ \phi = 180$), and we look at the far field diffracted in the plane (xOz): ($\theta = 0.360$, $\phi = 180$). On the right, we show the radar cross-section obtained. The first peak at $\theta = 225$ corresponds to the shadow zone, the second peak at $\theta = -45$ represents the reflected wave.

6. CONCLUSION

We have exposed the parallel FMM developed at INRIA and integrated in EADS integral equation software. The FMM is out of core, highly optimized, and very tunable. On workstations, we were able to solve industrial problems involving more than 1 million unknowns. In parallel, we have obtained very good results both on clusters of PC and supercomputers. Computations with up to 25 million unknowns were treated with success on 64-processors of IBM SP3. We plan to extend the FMM to more complex physical cases (including wires and dielectric), to convert post-treatments (such as far and close fields computations) to a multipole algorithm, and to develop solvers using all the features of FMM (especially the adjustable accuracy).

ACKNOWLEDGEMENTS

The computer time was provided by Centre Informatique National de l'Enseignement Supérieur, Montpellier, FRANCE (www.cines.fr).

REFERENCES

1. Rao SM, Wilton DR, Glisson AW. Electromagnetic scattering by surfaces of arbitrary shape. *IEEE Transactions on Antennas and Propagations* 1982; **30**(3):409–418.
2. Song JM, Lu CC, Chew WC, Lee SW. Fast Illinois solver code. *IEEE Transactions on Antennas and Propagations* 1998; **40**:3.
3. Sheng XQ, Jin JM, Song J, Chew WC, Lu CC. Solution of combined field integral equation using multilevel fast multipole algorithm for scattering by homogeneous bodies. *IEEE Transactions on Antennas and Propagations* 1998; **46**(11):1718–1726.
4. Coifman R, Rokhlin V, Wandzura S. The FMM for the wave equation: A pedestrian description. *IEEE Transactions on Antennas and Propagations* 1993; **35**(3):7–12.
5. Collino F, Millot F. La méthode multipôle à deux compsantes pour l'électromagnétisme. *Technical Report TR/EMC/01/22*, CERFACS, 2001.
6. Darve E. Méthodes multipôles rapides: Résolution des équations de Maxwell par formulations intégrales. *Thèse de doctorat de l'université*, Paris 6, Juin 1999.
7. Darve E. The fast multipole method 1: error analysis and asymptotic complexity. *SIAM Journal on Numerical Analysis* 2000; **38**(1):98–128.
8. Sylvand G. Méthode Multipôle Rapide en Electromagnétisme: Performances, Parallélisation, Applications. *Thèse de doctorat de l'Ecole Nationale des Ponts et Chaussées*, Juin 2002.
9. Frayssé V, Giraud L, Gratton S. A Set of GMRES routines for real and complex arithmetics. *Technical Report TR/PA/97/49*, CERFACS, 1997.

