Journal of Computational Physics **160,** 195–240 (2000)
doi:10.1006/jcph.2000.6451, available online at http://www.idealibrary.com on IDEAL®

The Fast Multipole Method: Numerical Implementation

Eric Darve

Center for Turbulence Research, Stanford University, Stanford, California 94305-3030
E-mail: darve@ctr.stanford.edu

Received June 8, 1999; revised December 15, 1999

We study integral methods applied to the resolution of the Maxwell equations where the linear system is solved using an iterative method which requires only matrix–vector products. The fast multipole method (FMM) is one of the most efficient methods used to perform matrix–vector products and accelerate the resolution of the linear system. A problem involving N degrees of freedom may be solved in $CN^{\text{iter}} N \log N$ floating operations, where C is a constant depending on the implementation of the method. In this article several techniques allowing one to reduce the constant C are analyzed. This reduction implies a lower total CPU time and a larger range of application of the FMM. In particular, new interpolation and anterpolation schemes are proposed which greatly improve on previous algorithms. Several numerical tests are also described. These confirm the efficiency and the theoretical complexity of the FMM. © 2000 Academic Press

Key Words: fast multipole method; electromagnetic theory; Scattering, Iterative Method, Matrix Compression Algorithms, Computational Aspects.

INTRODUCTION

Since the early papers by Rokhlin [21, 26, 50, 51], the fast multipole method (FMM) has proven to be a very powerful and efficient scheme for solving the Maxwell equation with an integral formulation (see, for example, [56–58]). However, implementation of the FMM proved to be a rather difficult task in part because of its complexity (many lines in a complex and long code) and because of the need to optimize all the steps of the FMM. Even though the asymptotic complexity is $O(N \log N)$, where N is the number of degrees of freedom, the constant in $O(N \log N)$ is large and thus several "tricks" must be used to reduce it (see, for example, the article by Bindiganavale and Volakis [9]). These are presented in this article.

The object of this paper is to give a general overview of the implementation of the FMM. It tries to address the different problems and options that someone willing to implement the FMM will encounter.

We would like to emphasize that the application of the FMM to Maxwell is very different from its application to the other fields, such as molecular dynamics simulation, N-body problem (for an introduction see [37, 42]), electrostatic or magnetostatic, Laplace equation, Hartree–Fock theory, Yukawa potential [3], etc. All these applications of the FMM deal with kernels like $1/|r - r'|^p$, where p is an integer $p \geq 1$ (compare to $\exp(\iota\kappa|r - r'|)/|r - r'|$ for the Maxwell equations). We now review some of these applications (see, for example, [30] to get a general view of all the applications). A variant of the FMM, called the continuous fast multipole method [62] was developed for example to solve Hartree–Fock problems and density functional calculations [14, 15, 45]. See also [13] for a recent description of the LinK algorithm. The reader will refer to [59] for a recent review on fast methods (including the FMM) for molecular dynamics simulation and to [44] for an application of the FMM with periodic boundary conditions. A significant effort was made in this area which led to several optimizations of the original algorithm (see [16–18, 34, 61]). However, it must be noted that these applications differ from electrodynamics applications for the following reasons: the expansion formulas and issues such as the total complexity of the computation are very different; the growth of the number of terms in the multipole expansion with the size of the cluster is specific to the electrodynamics; and moreover, there are specific issues related to the instability of the method when the electrical size of the smallest clusters goes to zero. The error estimations are also very different. We will refer the reader to [35, 36] for the particle system problem and to [22] for the Maxwell problem, etc. These differences can be seen in the fact that most of the applications of the FMM deal with kernels which are smooth functions with slow variations (regarding this concept see, for example, [27]). To the contrary, in this paper we deal with an oscillating kernel $\exp(\iota\kappa|r - r'|)/|r - r'|$.

The first section proposes an interpretation of the FMM in terms of plane wave superposition. This will make the physical interpretation of the FMM clearer. Various tricks, which can be found in [55], are then recalled, which allow one to reduce considerably the number of matrix–vector products to perform, in particular for the combined field integral equation.

The second section will address a key tool needed for the FMM: fast interpolation and anterpolation schemes. We will recall the currently available schemes and propose a variant of the sparse algorithm suggested by Lu and Chew in [43]. It reduces the complexity of the FMM to a minimum of $O(N \log N)$. The improvement of the new algorithm compared to Lu and Chew's original algorithm consists in an increased accuracy and a reduced constant for the complexity.

The third section will consider the ray propagation concept of Wagner and Chew [60] and will show the possible domain of application of this method. The main conclusion is that it should be used for the single step FMM but should perform poorly for the multistep scheme.

The fourth and fifth sections describe a few details of the implementation, regarding the number of poles to take at each level and the memory requirements. We propose an algorithm to reduce the memory by a factor of 2.

The last section is concerned with numerical tests. A scattering sphere is studied for several frequencies and an industrial test case is considered, called the CETAF. We compare our code, using the FMM, with an industrial code, SPECTRE, of Dassault-Aviation.

General notations:

$\imath$	$\sqrt{-1}$
λ	wavelength
κ	wave number $\kappa = 2\pi/\lambda$
∇_Γ	2D-divergence on the surface
$h_l^{(1)}$	spherical Hankel function of the first kind
P_m	Legendre polynomials
P_{ml}	associated Legendre functions
Z	$Z = 120\pi$

1. THE MULTILEVEL FAST MULTIPOLE METHOD

1.1. Plane Wave Approximation

Since many articles are devoted to the mathematical aspects of the FMM, we will skip the mathematical derivation of the algorithm. For that we refer the reader to previous articles ([21], for example, or [55]), which give the details of the method.

Here we have chosen to start with a physical interpretation of the FMM which will make clear to the reader the kind of approximation that is made with the FMM.

Let us consider a simple radiating point located at the origin $R(P) = e^{\imath\kappa r}/r$, where the observation point is $P = (r, \theta, \phi)$ in spherical coordinates.

The sphere is denoted by

$$S^2 = \{x \in \mathbb{R}^3/|x| = 1\}.$$

We denote in the following by $T_{l,P}(s)$ a function ("transfer" function) defined on S^2,

$$T_{l,P}(s) = \sum_{m=0}^{l} \frac{(2m+1)\imath^m}{4\pi} h_m^{(1)}(\kappa|P|)P_m(\cos(s, P)), \tag{1}$$

where the integer l is a truncation parameter, P is a vector of $\mathbb{R}^3$, and $\cos(s, P)$ is the cosine of the angle between s and P.

The FMM is based on the formula

$$\frac{e^{\imath\kappa|P+M|}}{|P+M|} = \imath\kappa \lim_{l\to+\infty} \int_{S^2} e^{\imath\kappa\langle s,M\rangle} T_{l,P}(s), \tag{2}$$

where S^2 is the sphere of radius 1 and $\langle\, \rangle$ is the scalar product. Equation (2) can be approximated, after discretizing the integral on S^2, by

$$\frac{e^{\imath\kappa|P+M|}}{|P+M|} \sim \imath\kappa \sum_k \omega_k e^{\imath\kappa\langle s_k,M\rangle} T_{l,P}(s_k),$$

where s_k are discrete directions of S^2 and ω_k quadrature weights. The idea behind the FMM is thus to approximate $R(P)$ locally by superposing a finite number of plane waves of direction of propagation s_k, of frequency $2\pi/\kappa$ and of complex amplitude $T_{l,P}(s_k)$. This is described in Fig. 2.

It is possible under certain hypothesis to simplify this expression and to point out very simple properties of $T_{l,P}(s)$.

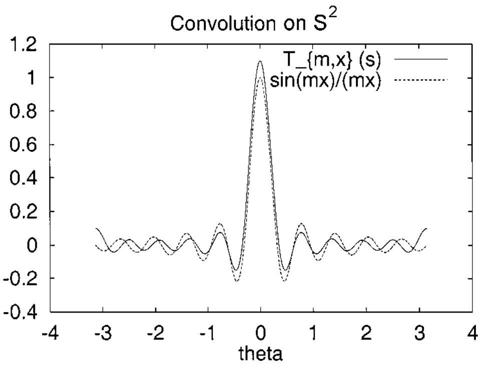

FIG. 1. Convolution on S^2.

If $\kappa|P| \gg 1$ we can use the asymptotic approximation

$$h_m^{(1)}(\kappa|P|) \sim \imath^{-m} \frac{e^{\imath\kappa|P|}}{\imath\kappa|P|}, \tag{3}$$

from which we can rewrite the integral as a convolution

$$\begin{aligned} \frac{e^{\imath\kappa|P+M|}}{|P+M|} &= \frac{e^{\imath\kappa|P|}}{|P|} \lim_{l\to+\infty} \int_{S^2} e^{\imath\kappa\langle s,M\rangle} \sum_{m=0}^{l} \frac{2m+1}{4\pi} P_m(\cos(s,\,P)) \\ &= \frac{e^{\imath\kappa|P|}}{|P|} \lim_{l\to+\infty} \frac{l+1}{4\pi} \int_{S^2} e^{\imath\kappa\langle s,M\rangle} \frac{P_l(\cos(s,\,P)) - P_{l+1}(\cos(s,\,P))}{1-\cos(s,\,P)}. \end{aligned} \tag{4}$$

The function

$$\frac{(2l+1)}{4\pi} \frac{P_l(\cos(s,\,P)) - P_{l+1}(\cos(s,\,P))}{1-\cos(s,\,P)}$$

behaves like $\sin(l(x-x_0))/x-x_0$ and converges to a Dirac function located at $s = P/|P|$ as $l \to +\infty$ (see Fig. 1).

Considering the remarks on (4), we proved that there are three regions of approximation. If P is far enough from the origin then $R(P)$ can locally be approximated by a single plane wave, with direction of propagation $P/|P|$. As P gets closer to the origin $R(P)$ will be approximated not by a single plane wave but rather by a superposition of several plane waves:

- $|P| \to +\infty$: one plane wave direction;
- $|P| \gg 1$: a few directions around $P/|P|$;
- $|P| = O(1)$: all directions have to be considered.

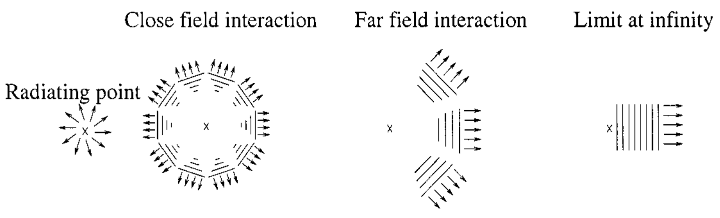

FIG. 2. Plane wave approximation.

In a multilevel implementation of the FMM, since the clusters never get very far from each other (relative to their size), we must consider all the discrete directions s_k to obtain a valid approximation. For a more complete description of those properties see Section 3.

1.2. Electric Field Integral Equation

We present the application of the FMM to the resolution of the EFIE (electric field integral equation).

The resolution of the EFIE with a Galerkin method leads to the resolution of $Ma = b$ with M and b defined by

$$M_{i,j} = \int_{\Gamma\times\Gamma} \frac{e^{\imath\kappa|x-y|}}{4\pi|x-y|}\left(\mathbf{J}_i(x)\mathbf{J}_j(y) - \frac{1}{\kappa^2}\nabla_\Gamma\mathbf{J}_i(x)\nabla_\Gamma\mathbf{J}_j(y)\right)d\Gamma(x)\,d\Gamma(y)$$
$$b_i = \frac{\imath}{\kappa Z}\int_\Gamma \mathbf{E}_t^{\mathrm{inc}}(x)\mathbf{J}_i(x)\,d\Gamma(x),$$

where $\mathbf{J}_i(x)$ and $\mathbf{J}_j(y)$ are the Rao–Wilton–Glisson basis functions (see [47]) and $\mathbf{E}_t^{\mathrm{inc}}$ is the tangential part of the incident field. See Fig. 3.

The resolution of this linear system with a dense complex matrix can be done with an iterative method (bi-conjugate gradient, GMRES, etc.). The CPU intensive part of the resolution becomes the computation of matrix–vector products with M.

We denote by M' the matrix

$$M'_{i,j} \overset{\mathrm{def}}{=} \frac{e^{\imath\kappa|x_i-x_j|}}{4\pi|x_i-x_j|}.$$

The FMM can be applied successfully to M'. Let us consider two clusters of points P_1 and P_2, of centers O_1 and O_2. For all points x_i in P_1 and x_j in P_2 we can accelerate the matrix–vector product using the plane wave expansion of $M'_{i,j}$,

$$M'_{i,j} = \sum_p \omega_p e^{\imath\kappa\langle s_p, x_i - O_1\rangle} T_{l,O_1-O_2}(s_p) e^{\imath\kappa\langle s_p, O_2 - x_j\rangle}, \tag{5}$$

where s_p are the quadrature points on S^2 and ω_p their weight. We will call $T_{l,O_1-O_2}(s_p)$ the "transfer function" associated to the pair of clusters (P_1, P_2).

We now explain how a product with M can be decomposed into products with M'.

If the edges i and j are far enough, $M_{i,j}$ is a regular integral and can be approximated by a numerical integration with Gauss quadrature points (typically three Gauss points per triangle).

We denote $\mathbf{e}_j^v$ a vector with three coordinates, where j corresponds to the Gauss point x_j, and e_j^s a scalar.

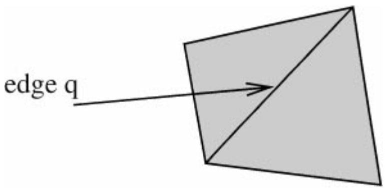

FIG. 3. Domain of definition of $J_q(x)$.

The product of M by a vector a is obtained by multiplying M' by $\mathbf{e}^v$ and by e^s

$$Ma \Rightarrow M'\mathbf{e}^v \qquad \text{and} \qquad M'e^s \tag{6}$$

with

$$\mathbf{e}^v_j \overset{\text{def}}{=} W_j \sum_r a_r \mathbf{J}_r(x_j) \tag{7}$$

$$e^s_j \overset{\text{def}}{=} \frac{\imath}{\kappa} W_j \sum_r a_r \nabla_\Gamma \mathbf{J}_r(x_j), \tag{8}$$

where the sum on r is a sum on the edges of the triangle containing x_j and W_j is the weight associated with the quadrature point x_j.

The vector $u = Ma$ can then be reconstructed from $\mathbf{f}^v \overset{\text{def}}{=} M'\mathbf{e}^v$ and from $f^s \overset{\text{def}}{=} M'e^s$ with the formula

$$u_i = \sum_r W_r \left(\left\langle \mathbf{J}_i(x_r), \mathbf{f}^v_r \right\rangle + \frac{\imath}{\kappa} \nabla_\Gamma \mathbf{J}_i(x_r)\, f^s_r \right),$$

where the sum on r is a sum on the Gauss points x_r for which $\mathbf{J}_i(x_r) \neq 0$.

1.3. MFIE and CFIE Formulation

Some modification have to be made in order to apply the FMM to the magnetic field integral equation (MFIE) and combined field integral equation (CFIE) formulations. Those modifications lead to an optimal implementation.

The MFIE is defined by

$$\frac{1}{2}\mathbf{J} \times \mathbf{n} + \left[\int_\Gamma \nabla_y \frac{e^{\imath\kappa|x-y|}}{4\pi|x-y|} \times \mathbf{J}(y)\, d\Gamma(y) \right]_t = \mathbf{H}^{\text{inc}}_t,$$

where $\times$ is the vectorial product and t denotes the tangential part of a vector. This equation leads to the following linear system, with a Galerkin method,

$$\frac{1}{2}\int_\Gamma \mathbf{J}_i(x) \cdot \mathbf{J}_j(x)\, d\Gamma(x) + \int_\Gamma d\Gamma(x) \mathbf{J}_i(x) \cdot \mathbf{n}(x) \times \int_\Gamma d\Gamma(y) \nabla_y \frac{e^{\imath\kappa|x-y|}}{4\pi|x-y|} \times \mathbf{J}_j(y)\, d\Gamma(y)$$
$$= \int_\Gamma d\Gamma(x) \mathbf{J}_i(x) \cdot \mathbf{n}(x) \times \mathbf{H}^{\text{inc}}_t(x),$$

where the scalar product is denoted by a dot.

To achieve a better convergence we use the CFIE, which is a linear combination of EFIE and MFIE defined by

$$\text{CFIE} = \alpha \text{EFIE} + (1-\alpha)\frac{i}{\kappa}\text{MFIE}.$$

The combined field integral equation has the advantage of having a much better condition number than both EFIE and MFIE. Moreover, an appropriate choice of α eliminates the convergence difficulties due to possible resonances on the scattering object. Thus it is the method of choice for numerical applications.

However, a closer look at MFIE and CFIE reveals that the FMM cannot be applied directly to those formulas or at least not in a straightforward way as for EFIE.

Considering Eq. (5), we can derive three other formula:

$$\frac{e^{\imath\kappa|x-y|}}{4\pi|x-y|} = \sum_p \omega_p e^{\imath\kappa\langle s_p, x-O_1\rangle} T_{l,O_1-O_2}(s_p) e^{\imath\kappa\langle s_p, O_2-y\rangle}$$

$$\nabla_x \frac{e^{\imath\kappa|x-y|}}{4\pi|x-y|} = \imath\kappa \sum_p \omega_p e^{\imath\kappa\langle s_p, x-O_1\rangle} T_{l,O_1-O_2}(s_p) e^{\imath\kappa\langle s_p, O_2-y\rangle} s_p$$

$$\nabla_y \frac{e^{\imath\kappa|x-y|}}{4\pi|x-y|} = -\imath\kappa \sum_p \omega_p e^{\imath\kappa\langle s_p, x-O_1\rangle} T_{l,O_1-O_2}(s_p) e^{\imath\kappa\langle s_p, O_2-y\rangle} s_p$$

$$\nabla_x \nabla_y \frac{e^{\imath\kappa|x-y|}}{4\pi|x-y|} = \kappa^2 \sum_p \omega_p e^{\imath\kappa\langle s_p, x-O_1\rangle} T_{l,O_1-O_2}(s_p) e^{\imath\kappa\langle s_p, O_2-y\rangle} s_p \otimes s_p.$$

Using these formulas we can derive a scheme adapted to EFIE, MFIE, and CFIE.

EFIE

We can reduce the number of matrix–vector products from four to a minimum of three with the following trick.

We denote by x_j the Gauss quadrature points on the surface of the scattering object.

Reduction to four matrix–vector products. Now we consider two clusters P and Q that are separated by at least one diameter. The sub-matrix $M'(P, Q)$ is defined by

$$M'(P,Q)_{i,j} = \begin{cases} M'_{i,j} & \text{if } x_i \in P \text{ and } x_j \in Q \\ 0 & \text{otherwise.} \end{cases}$$

We define $M(P, Q)$ in a similar way.

The product of $M'(P, Q)$ with a scalar quantity a'_j can be accelerated with the FMM. The algorithm can be decomposed in three steps:

The first step of the standard FMM consists in computing the radiation function F_Q defined on S^2 with the formula

$$F_Q(s_p) = \sum_{j/x_j \in Q} a'_j e^{\imath\kappa\langle s_p, O_Q - x_j\rangle},$$

where O_Q is the center of the cluster.

The second step (transfer) consists in the application of the transfer function

$$G_p(s_p) = T_{l,O_P-O_Q}(s_p) F_Q(s_p).$$

The final vector of the matrix–vector product is denoted by u'.

The last step of the standard FMM consists in an integration on S^2 for all Gauss points $x_i \in P$,

$$u'_i = \sum_p \omega_p e^{\imath\kappa\langle s_p, x_i - O_p\rangle} G_P(s_p).$$

With this method a matrix–vector product with $M(P, Q)$ is decomposed into four products with matrix $M'(P, Q)$, three products for the vectorial quantity $\mathbf{e}^v$ and one for e^s.

Reduction to three matrix-vector products. The first and last steps can be modified adequately. The trick to reduce the number of matrix–vector products is the following. We denote by $\mathbf{F}_Q(s_p)$ and $\mathbf{G}_P(s_p)$ two vectors defined for all directions $s_p \in S^2$. Then a matrix–vector product can be performed for the EFIE formulation with a modification of the first and last step only of the FMM:

1st step:

$$\mathbf{F}_Q(s_p) = \sum_r a_r \sum_j W_j e^{\imath\kappa\langle s_p, O_Q - x_j\rangle} (\mathbf{J}_r(x_j) - \langle s_p, \mathbf{J}_r(x_j)\rangle s_p). \tag{9}$$

The reader will compare this formula with the previous method where we had to compute

$$\mathbf{e}_j^v \overset{\text{def}}{=} W_j \sum_r a_r \mathbf{J}_r(x_j) \qquad e_j^s \overset{\text{def}}{=} \frac{\imath}{\kappa} W_j \sum_r a_r \nabla_\Gamma \mathbf{J}_r(x_j)$$

followed by

$$\sum_{j/x_j \in Q} e^{\imath\kappa\langle s_p, O_Q - x_j\rangle} \begin{cases} \mathbf{e}_j^v \\ e_j^s \end{cases}.$$

The sum on r in Eq. (9) is a sum on all the edges in cluster Q, while the sum on j is a sum on all the Gauss points for which $\mathbf{J}_r(x_j) \neq 0$. We recall that the Rao–Wilton–Glisson basis function $\mathbf{J}_r(x)$ has nonvanishing values *only* on the two triangles that have the edge r in common.

The second step of the FMM is applied to each component of $\mathbf{F}_Q(s_p)$ as before and we obtain a function $\mathbf{G}_P(s_p)$.

Last step: With this new formula for the FMM, the product $u = M(P, Q)a$ is obtained with

$$u_i = \sum_j W_j \sum_p \omega_p e^{\imath\kappa\langle s_p, x_j - O_P\rangle} \langle \mathbf{J}_i(x_j), \mathbf{G}_P(s_p)\rangle,$$

where the sum on p is a sum on all the directions $s_p \in S^2$ and where the sum on j is a sum on all the Gauss points for which $\mathbf{J}_i(x_j)$ is different from zero.

With this method a matrix–vector product with the FMM reduces to three matrix vectors with scalars: one for each component of $\mathbf{F}_Q$ and $\mathbf{G}_P$.

MFIE

With the same method we have to perform three matrix–vector products, with the following formula for the first and last step of the FMM:

1st step:

$$\mathbf{F}_Q(s_p) = \sum_r a_r \sum_j W_j e^{\imath\kappa\langle s_p, O_Q - x_j\rangle} \mathbf{J}_r(x_j).$$

Last step:

$$u_i = -\imath\kappa \sum_j W_j \sum_p \omega_p e^{\imath\kappa\langle s_p, x_j - O_P\rangle} \langle [(\mathbf{J}_i(x_j) \times \mathbf{n}(x_j)) \times s_p], \mathbf{G}_P(s_p)\rangle. \tag{10}$$

This formula is a consequence of

$$\begin{aligned}\langle \mathbf{J}_i(x_j), \mathbf{n}(x_j) \times (\mathbf{G}_P(s_p) \times s_p)\rangle &= \langle \mathbf{G}_P(s_p) \times s_p, \mathbf{J}_i(x_j) \times \mathbf{n}(x_j)\rangle \\ &= \langle s_p \times (\mathbf{J}_i(x_j) \times \mathbf{n}(x_j)), \mathbf{G}_P(s_p)\rangle \\ &= -\langle (\mathbf{J}_i(x_j) \times \mathbf{n}(x_j)) \times s_p, \mathbf{G}_P(s_p)\rangle\end{aligned}$$

since the term

$$\langle \mathbf{J}_i(x_j), \mathbf{n}(x_j) \times (\mathbf{G}_P(s_p) \times s_p)\rangle$$

corresponds to the computation of

$$\begin{aligned}&\int_\Gamma d\Gamma(x)\left\langle \mathbf{J}_i(x), \mathbf{n}(x) \times \int_\Gamma d\Gamma(y)\mathbf{J}_j(y) \times \nabla_x \frac{e^{\imath\kappa|x-y|}}{4\pi|x-y|}\, d\Gamma(y)\right\rangle \\ &\quad = \int_\Gamma d\Gamma(x)\left\langle \mathbf{J}_i(x), \mathbf{n}(x) \times \int_\Gamma d\Gamma(y)\nabla_y \frac{e^{\imath\kappa|x-y|}}{4\pi|x-y|} \times \mathbf{J}_j(y)\, d\Gamma(y)\right\rangle.\end{aligned}$$

Thus formula (10) corresponds to the MFIE formulation.

CFIE

We notice the fact that

$$\langle [(\mathbf{J}_i(x_j) \times \mathbf{n}(x_j)) \times s_p], s_p\rangle = 0,$$

from which we can deduce a scheme for the CFIE formulation.

1st step:

$$\mathbf{F}_Q(s_p) = \sum_r a_r \sum_j W_j e^{\imath\kappa\langle s_p, O_P - x_j\rangle}[(\mathbf{J}_r(x_j) - \langle s_p, \mathbf{J}_r(x_j)\rangle s_p)].$$

Last step:

$$u_i = \sum_j W_j \sum_p \omega_p e^{\imath\kappa\langle s_p, x_j - O_P\rangle}\langle \alpha\mathbf{J}_i(x_j) + (1-\alpha)[(\mathbf{J}_i(x_j) \times \mathbf{n}(x_j)) \times s_p], \mathbf{G}_P(s_p)\rangle.$$

The reader will refer to an article by Song and Chew for a similar scheme [53].

2. INTERPOLATION SCHEMES

Notations:

Legendre polynomials	$P_l(\cos\theta), l \geq 0$
associated Legendre functions	$P_{ml}(\cos\theta), -l \leq m \leq l, l \geq 0$
spherical harmonics	$Y_{ml}(\theta, \phi), -l \leq m \leq l, l \geq 0$

These functions are defined by

$$P_l(x) \overset{\text{def}}{=} \frac{1}{2^l l!} \frac{d^l (x^2-1)^l}{dx^l}$$

$$P_{ml}(x) = \sqrt{(2l+1)\frac{(l-m)!}{(l+m)!}}\,\bar{P}_{ml}(x) \overset{\text{def}}{=} (-1)^m \sqrt{(2l+1)\frac{(l-m)!}{(l+m)!}}(1-x^2)^{(1/2)m} \frac{d^m P_l(x)}{dx^m}$$

$$Y_{ml}(\theta,\phi) \overset{\text{def}}{=} \frac{1}{\sqrt{4\pi}} P_l^m(\cos\theta) e^{\imath m\phi}.$$

The set of functions Y_{ml} defines an L^2 orthonormal basis of $L^2(S^2)$. The functions $\bar{P}_{ml}$ are the usual associated Legendre functions. We normalized them so that

$$\int_{-1}^{1} (P_{ml}(x))^2\, dx = 1$$

Recursion Relations and Summation Formulas

We very briefly recall some very useful formulas:

$$\sqrt{l+m+1}P_{m+1,l}(x) = \frac{-1}{\sqrt{1-x^2}}\left\{\sqrt{l-m}\, x\, P_{ml}(x) - \sqrt{\frac{2l+1}{2l-1}(l+m)}\, P_{m,l-1}\right\}$$

$$\sqrt{\frac{(l+1+m)(l+1-m)}{2l+3}}P_{m,l+1} = \sqrt{2l+1}\, x\, P_{ml}(x) - \sqrt{\frac{(l-1-m)(l+m)}{2l-1}}P_{m,l-1}(x)$$

$$\sqrt{\frac{(l+m+1)(l+m)}{2l+3}}P_{m,l+1} = \sqrt{\frac{(l-m)(l-m+1)}{2l-1}}P_{m,l-1}(x) - \sqrt{2l+1}\sqrt{1-x^2}P_{m-1,l}(x).$$

The first terms needed to start the recursion are, for example,

$$P_{mm}(x) = (-1)^m \sqrt{\frac{2m+1}{2m}\frac{2m-1}{2m-2}\cdots\frac{3}{2}}\,(1-x^2)^{m/2}$$

$$P_{m,m+1}(x) = (-1)^m \sqrt{(2m+3)\frac{2m+1}{2m}\frac{2m-1}{2m-2}\cdots\frac{3}{2}}\,(1-x^2)^{m/2}x.$$

In the rest of the paper we will use the summation formula

$$(y-x)\sum_{m=0}^{n}(2m+1)P_m(x)P_m(y) = (n+1)(P_{n+1}(y)P_n(x) - P_n(y)P_{n+1}(x)).$$

2.1. Choice of Sample Points on S^2

There are several possibilities for the choice of sample points on S^2. The most straightforward is a uniform distribution of points:

$$\phi_i = \frac{2\pi}{I}i \qquad \text{and} \qquad \theta_j = \frac{\pi}{J}j.$$

This choice has the advantage of being simple but does not allow a very accurate integration of spherical harmonics. It requires twice as many points as with the Gauss–Legendre points.

A better choice are the Gauss–Legendre points. We want to integrate exactly the spherical harmonics, which are defined by

$$Y_{ml}(\theta, \phi) = e^{\imath m\phi} P_{ml}(\sin\theta).$$

The integral on S^2 is

$$\int_{S^2} e^{\imath m\phi} P_{ml}(\sin\theta) \cos\theta \, d\theta \, d\phi.$$

Thus it appears that if we compute this integral using Fubini's theorem

$$\int_0^{2\pi} d\phi \, e^{\imath m\phi} \int_0^{\pi} d\theta \, P_{ml}(\sin\theta) \cos\theta$$

the optimal choice of sample points are uniform points for ϕ_i and Gauss–Legendre points for θ_j. With this choice of points, we can integrate exactly all Y_{ml}, $-l \leq m \leq l$, $0 \leq l \leq L$, with $L+1$ points in the ϕ direction and $\frac{L+1}{2}$ points in the θ direction.

There is a simple proof of this fact. For $m \neq 0$ the discrete integral $\sum_l \frac{2\pi}{L+1} e^{\imath m\phi_l}$ will be equal to zero. Thus independently of the choice of points θ_j the integration is exact for all $m \neq 0$. Now for $m = 0$, we can rewrite

$$\int_0^{\pi} d\theta \, P_l(\sin\theta) \cos\theta = \int_{-1}^{1} P_l(x) \, dx$$

and the integration is exact if we use Gauss–Legendre points since P_l is a polynomial (the P_{ml} are in general not polynomials).

However, even better choices can be made with respect to the number of directions on S^2. One of these is explained in [44]. This optimal choice of samples on S^2 is guided by considerations on set of points invariant under groups of rotations. For the circle, in the case of unit weight function, the optimal choice of points is evenly spaced points. In the case of the sphere McLaren considers sets of points invariant under one of the finite groups of rotations associated with regular solids.

If g is the order of the group, the use of an invariant formula reduces by a factor of g the number of independent surface harmonics for which the formula must be made exact. This is proved in [44]. The most spectacular result is a formula with 72 points integrating exactly 255 spherical harmonics. McLaren also gives a sequential procedure for obtaining formulae with as many points as needed (Section 7.2 in his paper).

However, such a choice of points is much more difficult to code and leads to increased difficulties when trying to obtain a fast interpolation scheme using those points. As we will see, this is much easier with the Gauss–Legendre points.

2.2. Interpolation Algorithms

There are several interpolation algorithms available. We will discuss here the scheme due to Alpert and Jakob–Chien [5] and the one due to Song, Lu, and Chew [40, 41, 53].

Another possible scheme is the Lagrange interpolation for which simple error analysis can be done. The mathematical analysis shows that the convergence is exponential as the number of points on S^2 increases (see [40]). It should be noted that we interpolate functions which have a finite bandwidth in the Fourier space. Thus optimal efficiency can only be attained if we take advantage of this property. More general interpolation algorithms such as Lagrange interpolation or B-splines are not optimal in this case.

Finally, with uniformly spaced samples a few other algorithms are available, such as in [11, 12]. These use the approximate prolate spheroid series (see [38]) and the sampling window (Chebyshev sampling series; see [39]).

2.3. Sparse Interpolation

When the size of the problem becomes significant (above 100,000 degrees of freedom) the interpolations and anterpolations which have to be performed to transfer the information from the lower level to the upper level and vice versa become significant in terms of CPU. Several levels (between 5 and 10 for larger cases) are needed and the upper ones involve a larger and larger number of points on S^2 (hundreds of directions θ and ϕ). This problem is specific to the Maxwell case. Applications of the FMM to other areas (molecular dynamics, electrostatic, N-body problem, etc.) do not have this problem.

We now describe the sparse approximation scheme (Lu and Chew's scheme) used for the interpolations and the anterpolations.

Position of the Problem

We denote by f a function defined on S^2 and by s a point of S^2.

We will call f a function that is band-limited of degree K if

$$f(\theta, \phi) = \sum_{m,l \le K} f_{ml} Y_{ml}(\theta, \phi).$$

The set of functions Y_{ml} have the interesting property of uniform resolution on S^2. This means that any rotation of f can be represented by the same expansion with appropriately transformed coefficients. Thus even though the points on S^2 have a nonisotropic distribution the property of uniform resolution of Y_{ml} ensures that the FMM is an isotropic scheme.

We denote by (θ_k, ϕ_k), $1 \le k \le K$, a set of quadrature points on S^2 and by ω_k their weights, such that they integrate exactly all the spherical harmonics of degree inferior to $2K$. We denote by f_k, $1 \le k \le K$, the value of f at each node and by $f_{k'}$ its value at the interpolated nodes $(\theta'_{k'}, \phi'_{k'})$, $1 \le k' \le K'$.

The integer K' is supposed to be *proportional* to K (for the FMM $K' = 2K$). This hypothesis is important in order to evaluate the total complexity of the algorithm.

The interpolation or anterpolation consists in performing the following linear transformation on f_k,

$$f_{k'} = \sum_{m,l \le K} Y_{ml}(\theta'_{k'}, \phi'_{k'}) \sum_k Y^*_{ml}(\theta_k, \phi_k) \omega_k f_k,$$

for all k', where Y^*_{ml} denotes the complex conjugate. If we define the following rectangular matrix A of size $K' \times K$,

$$A_{k'k} = \sum_{m,l \le K} Y_{ml}(\theta'_{k'}, \phi'_{k'}) Y^*_{ml}(\theta_k, \phi_k)\omega_k,$$

then the interpolation/anterpolation is a simple matrix–vector product

$$f_{k'} = \sum_k A_{k'k} f_k.$$

However, the complexity of this transform is $K' \times K$. Such a scheme would result in an N^2 complexity for the total FMM, where N is the number of degrees of freedom. Contrary to the other applications of the FMM, the interpolation and anterpolation algorithms are thus crucial in the Maxwell (Helmholtz) case to obtain an $N \log N$ algorithm.

Possible Approaches

Three approaches to reducing this complexity are possible.

- The semi-naive scheme is the most straightforward and is the one from which Alpert and Jakob-Chien started their analysis in [5]. The semi-naive scheme consists in performing a forward FFT, a dense matrix–vector product on the θ angles, and a backward FFT. This scheme will be preferred when the size of the problem is small. It has a greater asymptotic complexity ($K^{1.5}$) but performs well for small clusters. Moreover this algorithm is exact (in exact arithmetic).
- The scheme due to Alpert and Jakob-Chien is described in [5]. The first step, as before, is a forward FFT on ϕ. Then the dense matrix corresponding to the θ angles is compressed and expedited with a 1D-FMM. Yarvin and Rokhlin [62] present a generalization of the 1D-FMM applicable to Alpert and Jakob-Chien's interpolation algorithm. See also [25, 61]. Finally a backward FFT is performed on ϕ. Its complexity is $O(K \log K)$. This is an approximate algorithm.

There are other schemes with similar complexity: see [23, 24].

- The last scheme is the one used in the MLFMA variant of the FMM due to Chew, Lu, and Song and is used in FISC. It consists in a sparsification of A. We now describe in greater details this scheme since our new scheme is a variant of this one. Its complexity is $O(K)$.

The idea of the sparsification comes from the 1D Fourier transform. If we replace the spherical harmonics by an exponential function the expression of A is replaced by

$$\begin{aligned} B_{k'k} &= \frac{2\pi}{K} \sum_{-K \le l \le K} e^{\imath l \frac{2\pi}{K'} k'} e^{-\imath l \frac{2\pi}{K} k} \\ &= \frac{2\pi}{K} \frac{e^{\imath 2\pi (K+1)\left(\frac{k'}{K'} - \frac{k}{K}\right)} - e^{-\imath 2\pi K\left(\frac{k'}{K'} - \frac{k}{K}\right)}}{e^{\imath 2\pi\left(\frac{k'}{K'} - \frac{k}{K}\right)} - 1} \\ &= \frac{2\pi}{K} \frac{\sin \pi(2K+1)\left(\frac{k'}{K'} - \frac{k}{K}\right)}{\sin \pi\left(\frac{k'}{K'} - \frac{k}{K}\right)}. \end{aligned}$$

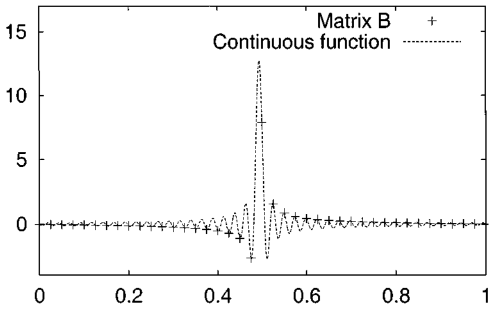

FIG. 4. Interpolation matrix.

This function shows a peak around $\frac{k}{K} \sim \frac{k'}{K'}$ and is decreasing like $(\frac{k}{K})^{-1}$ when $|\frac{k'}{K'} - \frac{k}{K}| \gg 1$. The width of the peak is of order $O(\frac{1}{K})$. Figure 4 illustrates the value of $B_{k'k}$ for each k and a given k'. We plot the discrete values of $B_{k'k}$ and a continuous function obtained by replacing the discrete value $\frac{k}{K}$ by a continuous variable $x \in [0, 1]$.

Thus it is possible to sparsify the matrix $B_{k'k}$ and remove all the coefficients smaller than a given threshold.

Let us now assume that we take a constant number of samples per period. Then for a given accuracy $\epsilon > 0$, the number of entries per line with modulus greater than ϵ is equal to the number of samples in a segment of length $\frac{C(\epsilon)}{K}$ (width of the peak) centered at $\frac{k'}{K'}$. Since the total number of sample points in $[0,\ 2\pi]$ is proportional to K, the number of entries per line greater than ϵ is of order $C(\epsilon)$ and thus is a constant independent of K or K'.

The same decrease can be observed for $A_{k'k}$ and thus it is possible to construct a sparse approximation $\tilde{A}_{k'k}$ of $A_{k'k}$. With the same type of argument as with $B_{k'k}$ we prove that the number of entries in $\tilde{A}_{k'k}$ with modulus greater than ϵ is equal to $C(\epsilon)$, where C is a function of ϵ only.

This scheme reduces the asymptotic complexity from $O(K \log K)$ to a minimum of $O(K)$.

The total complexity of the FMM is reduced from $O(N \log^2 N)$ to $O(N \log N)$.

2.4. New Sparsifying Algorithm

It is possible to greatly improve the sparsification of the matrix. A decrease in $1/x$ as observed with $B_{k'k}$ means that a significant number of points will have to be considered to achieve a good accuracy (10^{-2} relative error for example). Thus the compression factor for $A_{k'k}$ will not be so good.

This slow decrease is due to a discontinuity in the Fourier domain. Multiplying by $A_{k'k}$ is equivalent to computing a forward Fourier transform, then multiplying by the characteristic function $\mathbf{1}_{[-K,K]}$ defined by

$$\mathbf{1}_{[-K,K]}(k) = \begin{cases} 1 & \text{If } |k| \leq K \\ 0 & \text{otherwise} \end{cases}$$

and performing a backward Fourier transform.

The slow decrease is due to the discontinuity of $\mathbf{1}_{[-K,K]}(k)$. It is possible to achieve a faster decrease by choosing a function that goes smoothly from 0 to 1 and from 1 to 0 (see [19] for a similar idea applied in a different context).

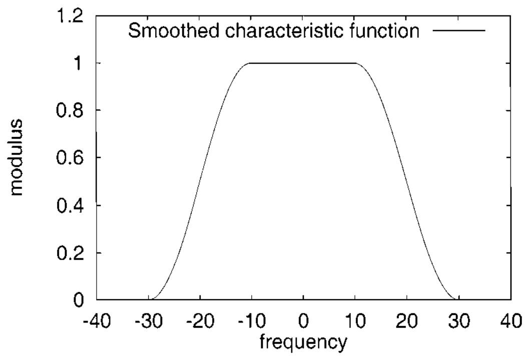

FIG. 5. Smoothed characteristic function, $\omega_1 = 15$, $\omega_2 = 30$.

We now define such a function. We denote by ω_1 and ω_2 two positive numbers such that $\omega_1 \le \omega_2$.

Let us compute the continuous Fourier transform of

$$C(\omega) = \begin{cases} \cos^2\left(\frac{(\omega+\omega_1)\pi}{2(\omega_2-\omega_1)}\right) & -\omega_2 \le w \le -\omega_1 \\ 1 & -\omega_1 \le \omega \le \omega_1; \\ \cos^2\left(\frac{(\omega-\omega_1)\pi}{2(\omega_2-\omega_1)}\right) & \omega_1 \le w \le \omega_2 \end{cases}$$

see Fig. 5.

This Fourier transform can be computed easily since $C(\omega)$ can be decomposed into a sum of trigonometric functions; the final expression for the Fourier transform is

$$\begin{aligned} \int d\omega C(\omega)\, e^{\imath\omega x} &= \frac{1}{(1-\tau^2)x}[\sin(\omega_2 x) + \sin(\omega_1 x)] \\ &= \frac{\cos\frac{\pi}{2}\tau}{1-\tau^2} \cdot \frac{2}{x} \sin\frac{\omega_1+\omega_2}{2}x \\ \tau &= \frac{x(\omega_2-\omega_1)}{\pi}. \end{aligned}$$

The Fourier transform of the characteristic function $\mathbf{1}_{[-\omega_1,\omega_1]}$ is

$$\frac{2}{x}\sin\omega_1 x.$$

The factor $\cos\frac{\pi}{2}\tau/(1-\tau^2)$ is thus the one responsible for a faster decrease at infinity.

When $\tau \ll 1$,

$$\int d\omega C(\omega)\, e^{\imath\omega x} \sim \frac{2}{x}\sin\frac{\omega_1+\omega_2}{2}x.$$

When $\tau \gg 1$,

$$\int d\omega C(\omega)\, e^{\imath\omega x} \sim \frac{\omega_1-\omega_2}{\pi\tau^3}(\sin(\omega_2 x) + \sin(\omega_1 x)).$$

With this smoothed characteristic function we achieve a decrease in $1/x^3$ which improves on the previous $1/x$.

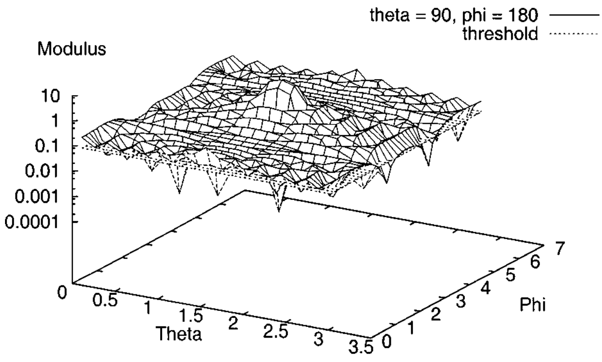

FIG. 6. Original scheme: log-scale on the z-axis.

We can now define a new interpolation matrix with improved sparsity. Let us denote by rK the bandwidth of our Fourier window. The new interpolation matrix is

$$\bar{A}_{k'k} = \sum_{m,l \le \pi K} C_l^{K,\pi K} Y_{ml}(\theta'_{k'}, \phi'_{k'}) Y^*_{ml}(\theta_k, \phi_k)\omega_k \tag{11}$$

$$C_l^{K,\pi K} = \begin{cases} 1 & 0 \le l \le K \\ \cos^2\left(\frac{(l-K)\pi}{2K(\pi-1)}\right) & K \le l \le \pi K. \end{cases} \tag{12}$$

Figures 6, 7, 8, and 9 correspond to the new interpolation scheme. The results for the anterpolation scheme are strictly equivalent, the anterpolation matrix being the transpose of the interpolation matrix.

To illustrate the improvement, we plot the modulus of the entries in a given line of $A_{k'k}$ and $\bar{A}_{k'k}$: see Figs. 6, 7, 8, and 9. The line number k' corresponds to the interpolated point (θ', ϕ') with $\theta' = 90$ and $\phi' = 180$. The θ angle is on the x-axis while the ϕ angle is on the y-axis. The modulus of the entries is on the z-axis. This clearly shows that the coefficients have a much faster decrease, which results in a greater sparsity. The threshold corresponds to a relative error of 10^{-2}.

The second figure (see Figs. 10 and 11) shows the convergence rate of the error in the approximate interpolation as a function of the number of non-zero entries per line in $A_{k'k}$

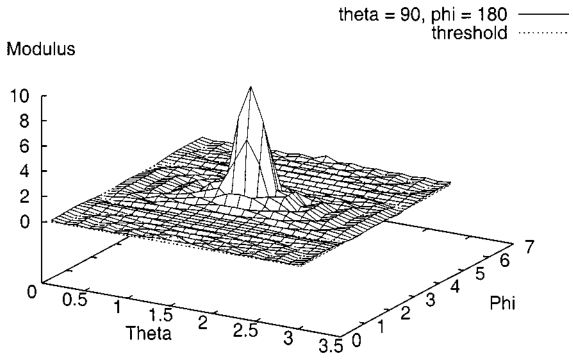

FIG. 7. Original scheme.

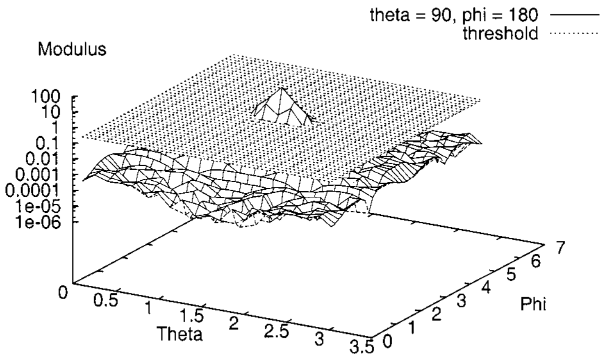

FIG. 8. New improved scheme: log-scale on the z-axis.

and $\overline{A}_{k'k}$. The number of non-zero entries is on the abscissa. This proves that much better accuracy can be achieved at a lower CPU cost.

Finally we present a comparison with the semi-naive scheme (with complexity $O(K^{1.5})$). Figure 12 shows that the semi-naive scheme is faster when the clusters are small and that the "sparse matrix" scheme should be used for the larger clusters. The integer n is the number of non-zero entries per line. With $n = 9$ interpolation points, the "sparse matrix" scheme is always faster than the semi-naive scheme. For $n = 25$ the cross point can be situated around 33. This corresponds to a cluster of diameter 10 wavelengths. For $n = 49$ the cross point is around 87 (diameter 28 wavelengths).

Moreover, it should be noted that, with a fixed number of interpolation points per line, the error slowly decreases with the size of the problem. Thus for a given accuracy the number of points should be reduced as the size of the problem grows. See Fig. 13 for an illustration. For example, for a relative error of 1%, the semi-naive scheme should be used for $K \leq 35$ (11 wavelengths); then the sparse scheme should be used with only nine interpolation points. For a 10% relative error the sparse scheme is even more efficient and can be used as soon as $K \geq 8$ (2.5 wavelengths).

Conclusion of the Section

This improved scheme is a variant of the scheme proposed by Chew, Lu, and Song. It does not change the asymptotic complexity $(O(K))$ but reduces its constant. This means

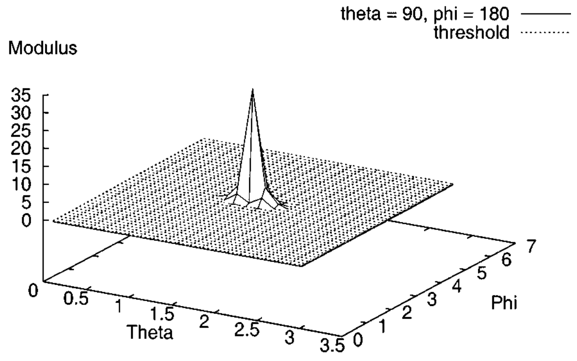

FIG. 9. New improved scheme.

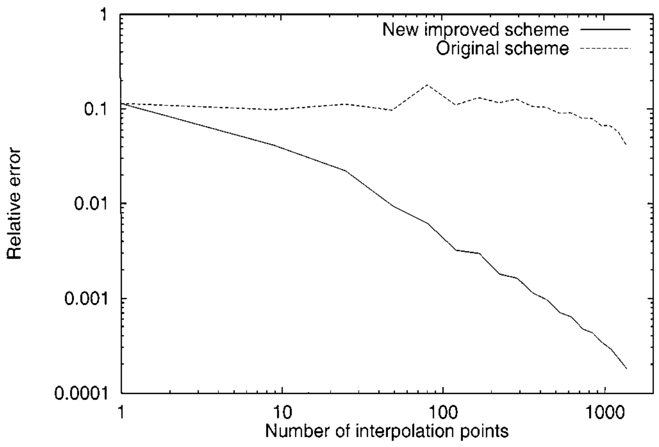

FIG. 10. Convergence rate of the approximate interpolation scheme: log-scale.

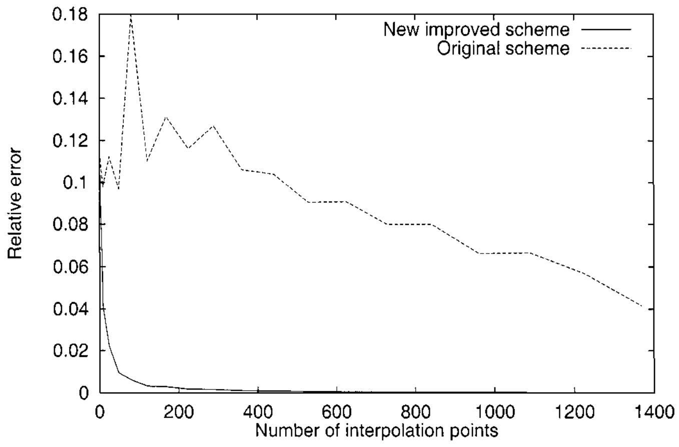

FIG. 11. Convergence rate of the approximate interpolation scheme.

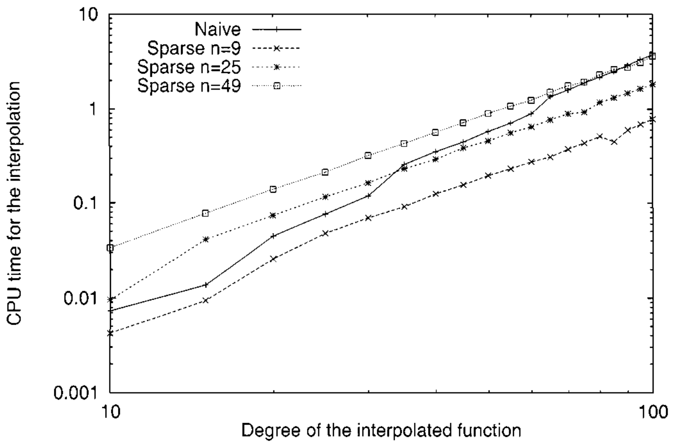

FIG. 12. Comparison with the semi-naive scheme.

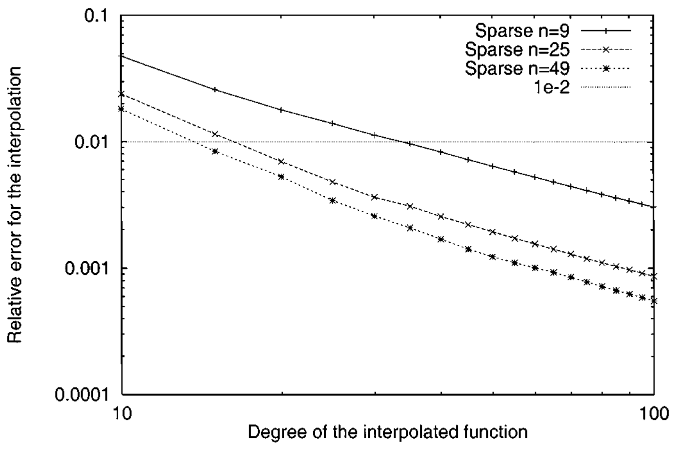

FIG. 13. Relative error for the "sparse scheme."

that the cross point with the semi-naive scheme will be lower and that it will result in an overall faster implementation of the FMM.

3. TRANSFER FUNCTIONS–TRANSFER LIST

Construction of the Transfer List

We now shortly describe the construction of the transfer list. This is a recursive construction where two lists must be built: one corresponding to the close clusters and the other corresponding to the clusters for which a transfer must be performed. Those two lists should be built simultaneously. See Algorithms 1 and 2.

Optimizing the Construction of the Transfer Functions

A possible optimization can be done regarding the number of transfer functions that have to be computed. Since the complexity involved in computing a transfer function is

ALGORITHM 1
Construction of the Transfer List

```
FUNCTION RecursiveBuild(Father)
for all CloseCluster, close to Father do
   for all NeighborhoodCluster, son of CloseCluster do
      for all CurrentCluster, son of Father do
         if NeighborhoodCluster is close to CurrentCluster then
            Store NeighborhoodCluster in the list CurrentCluster.ListClose
         else
            Store NeighborhoodCluster in the list CurrentCluster.ListTransfer
         end if
      end for
   end for
end for
for all CurrentCluster, son of Father do
   RecursiveBuild(CurrentCluster)
end for
End of FUNCTION
```

ALGORITHM 2
Main Program

```
1: FUNCTION main( )
2: CurrentCluster = TopCluster
3: Store CurrentCluster in the list CurrentCluster.ListClose
4: RecursiveBuild(CurrentCluster)
```

much higher than the complexity involved in applying the transfer vector to a cluster, their recomputation should be avoided as much as possible.

This can be optimized if the clusters are built using an oct-tree, where a cube is subdivided in height half-smaller cubes, in a recursive way. If the clusters are constructed this way, then, for a given level in the tree, the number of possible transfer vectors $O_P - O_Q$ is reduced to a minimum of 316 different vectors $(7 \times 7 \times 7 - 3 \times 3 \times 3)$. Thus an optimized algorithm would perform the transfers in the following way: see Algorithm 3.

This algorithm will be especially efficient at the lower levels in the tree for which the number of transfers to be performed is much larger than $316 (= 7 \times 7 \times 7 - 3 \times 3 \times 3)$.

Sparsifying the Transfer Matrix

As first suggested by Wagner and Chew in [58] it is, under certain conditions, possible to sparsify efficiently the translation matrix. This fact should be connected with our observations in Section 1.1 regarding the number of plane waves needed to approximate the radiated field. It has been shown that when the clusters are sufficiently far away from each other the radiated field can be approximated by superposing a limited number of plane waves with directions close to $(O_P - O_Q)/|O_P - O_Q$. Figure 1 shows that the transfer function $T_{l,O_P-O_Q}(s_k)$, $1 \le k \le K$ has maximal values around

$$s_k \sim s_{\text{ray}} \overset{\text{def}}{=} \frac{O_P - O_Q}{|O_P - O_Q|}$$

and then decreases when s_k is far from s_{ray}.

The expression for the transfer function is

$$T_{l,O_P-O_Q}(s) = \sum_{m=0}^{l} \frac{(2m+1)\imath^m}{4\pi} h_m^{(1)}(\kappa|O_P - O_Q|) P_m(\cos(s, O_P - O_Q))$$

(see Eq. (1) for the notations). The number of terms l in this series is related to the diameter

ALGORITHM 3
Algorithm to Reduce the Number of Transfer Vectors Computed

```
1: for all TVector, a transfer vector do
2:     for all (P,Q), pair of clusters such that $O_P - O_Q = $ TVector do
3:         Apply TVector to pair (P,Q)
4:     end for
5: end for
```

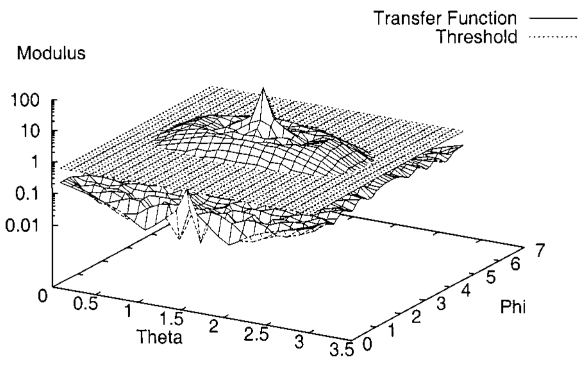

FIG. 14. $r = 1.1$, sparse transfer vector.

of O_P and O_Q,

$$l \sim \kappa \frac{\text{Diam}(O_P) + \text{Diam}(\text{O}_\text{Q})}{2},$$

where “Diam” is the diameter.

We proved that the approximation (3) was valid under the assumption $l \ll \kappa |O_P - O_Q|$. Thus we expect that the compression rate will depend on the ratio of the distance between the clusters to their size, denoted by r:

$$r \overset{\text{def}}{=} \frac{\text{Diam}(O_P) + \text{Diam}(O_Q)}{2|O_P - O_Q|}.$$

Figures 14–19 illustrate this fact. The relative error is set as before at 10^{-2} (threshold). The figures on the right correspond to the usual transfer vector. The peak centered around $\theta = 90^\circ$ and $\phi = 180^\circ$ becomes sharper as the distance between the clusters increase. Thus fewer terms have to be computed, which results in faster transfers between clusters and a lower complexity for the total FMM.

As before, we can increase the sparsity of the transfer vectors by using a smoothed characteristic function, which allows a smooth decay from 1 to zero. This is illustrated by the figures on the left. They correspond to a transfer vector with improved sparsity defined

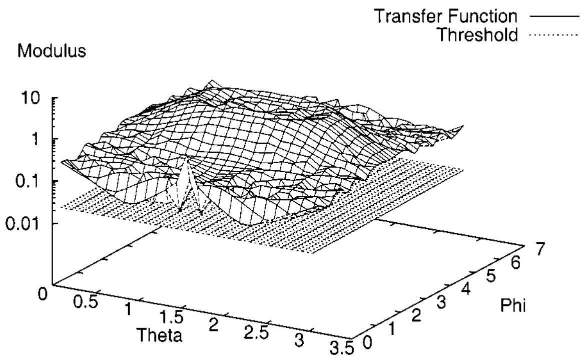

FIG. 15. $r = 1.1$, original transfer vector.

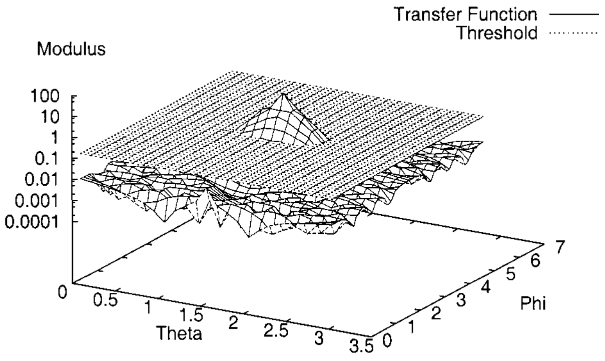

FIG. 16. $r = 3.0$, sparse transfer vector.

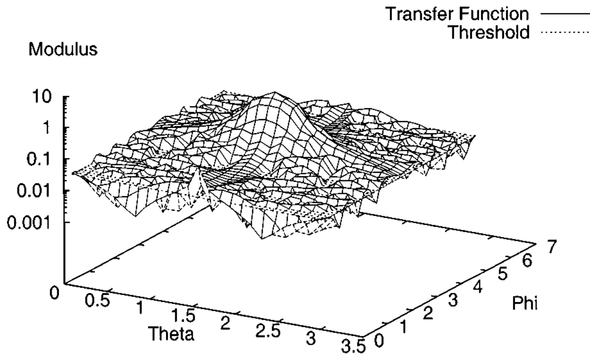

FIG. 17. $r = 3.0$, original transfer vector.

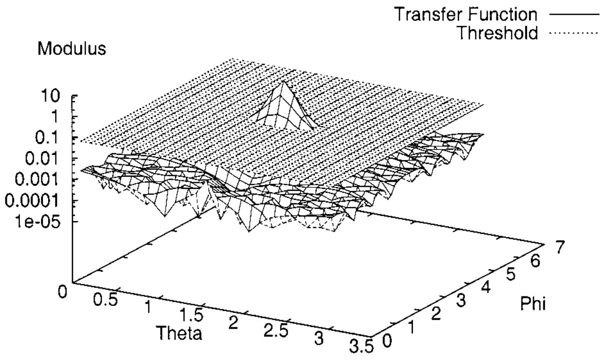

FIG. 18. $r = 6.0$, sparse transfer vector.

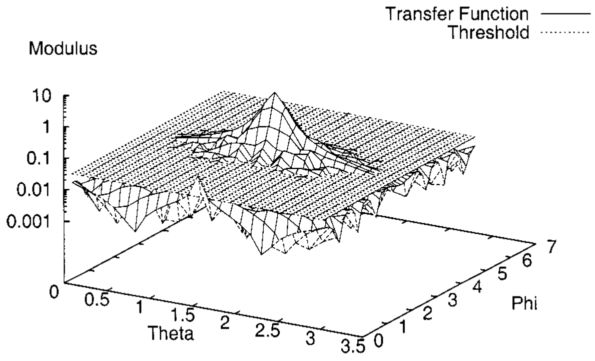

FIG. 19. $r = 6.0$, original transfer vector.

by

$$T^{\text{new}}_{l,O_P-O_Q}(s) = \sum_{m=0}^{l} \frac{(2m+1)\imath^m}{4\pi} h_m^{(1)}(\kappa|O_P - O_Q|)P_m(\cos(s, O_P - O_Q))$$
$$+ \frac{(2l+1)\imath^l}{4\pi} h_l^{(1)}(\kappa|O_P - O_Q|) \sum_{m=l+1}^{l'} C_m^{l,l'} P_m(\cos(s, O_P - O_Q)),$$

where $C_m^{l,l'}$ is defined in (12). The reader will notice that we do not consider Hankel functions with indices superior to l. This is due to the fact that the Hankel functions diverge when $l \gg \kappa|O_P - O_Q|$. This formula ensures a smooth decay from 1 to 0 and thus improves considerably the compression rate of the transfer vector. For the figures the integer l' was chosen equal to $2l$.

From these numerical experiments we can observe that the number of interpolation points decrease like $1/|O_P - O_Q|$ when $|O_P - O_Q|$ increases. Thus the number of non-zero elements per line in the sparsified matrix decreases like $1/|O_P - O_Q|^2$.

Remark. Since the total number of samples on the sphere is of order l^2 and since there is a decay in $1/|O_P - O_Q|^2$, we can deduce that the number of degrees of freedom in the interaction between the two clusters is of order r^2. This fact should be connected with similar results obtained by [28], later refined by [10, 29]. The concept of degrees of freedom was used successfully by Michielssen and Boag with the construction of the matrix decomposition algorithm (see [45, 46]).

Conclusion of the Section

In a multilevel FMM, the ratio r is close to 1 and thus only a poor compression rate can be achieved. This limits strongly the usefulness of this sparsifying scheme in a multilevel implementation.

However, the compression rate improves greatly when r becomes large compared to 1. In a one-level FMM, distances between clusters may become large (relative to their size). Thus this scheme can be used successfully with a one-level FMM, reducing considerably the time spent in computing and applying the transfer vectors for pairs of clusters very far away from each other.

4. NUMBER OF POLES IN THE MULTIPOLE EXPANSION

4.1. Convergence Test

A convergence criterion is necessary to determine where the series

$$(2l+1)\, j_l(v) h_l^{(1)}(u) P_l(\cos\gamma) \tag{13}$$

should be truncated. The notations are j_l, spherical Bessel function; v, diameter of the cluster; $h_l^{(1)}$, spherical Hankel function; u, distance between the cluster; P_l, Legendre polynomials.

Several formulas can be used to evaluate the number of poles needed in the multipole expansion:

1. Formula found in previous papers (see [50, 54]). A semi-empirical formula is used,

$$l = v + C\log(\pi + v), \tag{14}$$

where C is some constant and where v is the argument of the Bessel function j_l. The positive number v is equal to the diameter of the cluster.

2. Since this formula is actually a test on the convergence of the function j_l, we might as well consider the modulus of j_l, which yields the criterion that

$$|j_l(v)| \le \epsilon \qquad \text{and} \qquad l \ge v.$$

3. Finally we may test the convergence of the series, which yields the criterion that

$$\left|(2l+1)j_l(v)h^{(1)}(u)\right| \le \epsilon \qquad \text{and} \qquad l \ge v.$$

These different tests when tried numerically gave about the same results, as far as the CPU time/precision is concerned: Fig. 20. The figure was obtained by choosing different ϵ and C for the three methods. These tests correspond to a scattering sphere of radius 1. The wavelength of the incident wave is equal to 0.7. There are 3072 edges and 1026 points, and the FMM has four levels. The CPU time (y-axis) corresponds to the resolution of the linear system with an iterative method (GMRES) and a CFIE formulation. The precision (x-axis) corresponds to the relative error for the solution, defined as the relative difference between the FMM and the standard method (full assembly of the CFIE matrix, resolution with the same iterative method). Thus it measures exactly the error introduced by the FMM.

However, the figure indicates clearly that to maximize the precision while keeping the CPU time at a minimum, the parameters should be set to

1. $C = 2.25$.
2. $\epsilon = 0.001$.
3. $\epsilon = 0.66$.

With these parameters a relative error of 6×10^{-3} can be achieved. These results depend on the minimal size chosen for the clusters, which we set, for these particular numerical tests, at $\kappa.d = 1$, where d is the size of the cubic cluster (length of the side).

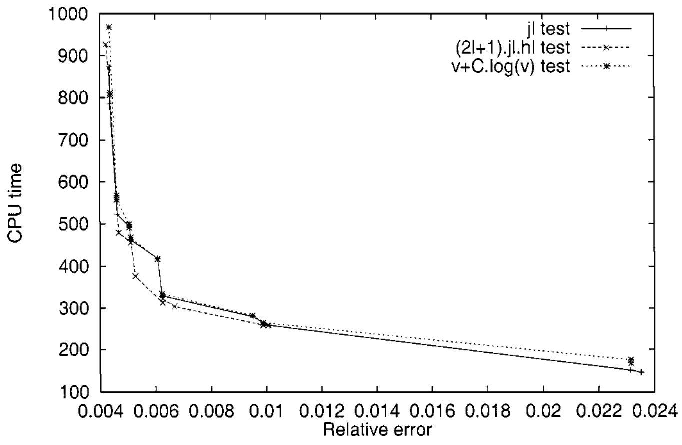

FIG. 20. Convergence tests.

4.2. Number of Samples on S^2

The previous subsection determined the number of terms l to consider in the expansion. However, it remains now to evaluate the number of samples needed to integrate exactly the radiation functions. The position of the problem is the following:

As a simplification we take $l = v$ as an approximation of the previous formula $v + C \log(\pi + v)$. We consider cluster P and its radiation function $F_P(s_k)$. The function $F_P(s_k)$ has a bandwidth of $l/2$ since the diameter of the cluster is equal to $v = l$. Then $F_P(s_k)$ is multiplied by the transfer function $T_{l,O_P-O_Q}(s_k)$, which has a bandwidth equal to l. Finally the resulting radiation function $G_Q(s_k)$ is multiplied by $e^{\iota\kappa\langle s_k, O_{Q'}-O_Q\rangle}$ (bandwidth $l/4$) and anterpolated to obtain the transfer function $G_{Q'}(s_k)$ associated with cluster Q', one level below cluster Q. The bandwidth of $G_{Q'}(s_k)$ being equal to $l/4$ (the diameter of the cluster is $v/2 = l/2$), we will need to anterpolate the first $l/4$ Fourier frequencies. See Fig. 21.

The final bandwidth the function to be integrated is

$$\frac{l}{2} + l + \frac{l}{4}\left(e^{\iota\kappa\langle s_k, O_{Q'}-O_Q\rangle} \text{ factor}\right) + \frac{l}{4} \text{ (anterpolation scheme)}.$$

Thus the total bandwidth is simply $2l$. For this level, the number of ϕ will be $2l + 1$, and the number of θ directions $(2l + 1)/2 \le l + 1$.

However for the interpolation scheme the bandwidth is much lower. The bandwidth of the function to be integrated is

$$\frac{l}{2} \text{ (bandwidth of } F_P(s_k)) + \frac{l}{2} \text{ (interpolation scheme)}.$$

Thus the total bandwidth is only l. No additional sample points are needed for the interpolation.

If we use the new sparsifying scheme for the anterpolation, then with an anterpolation function of bandwidth $rl/4$ (width of the window in the Fourier domain; see Eq. (12) for the notations), the bandwidth of the function which has to be integrated exactly is

$$\frac{l}{2} + l + \frac{l}{4} + \frac{rl}{4} = 2l\left(1 + \frac{r-1}{8}\right).$$

Thus a good compromise must be found between the compression rate, which requires larger r, and the total number of samples, which grows with r. The choice $r = 3$ seems to be a good choice; it slightly increases the number of samples, $3l^2$ instead of $2l^2$, while maintaining an almost optimal compression rate. This choice was made in the figures illustrating the new sparsifying scheme (Section 2.4).

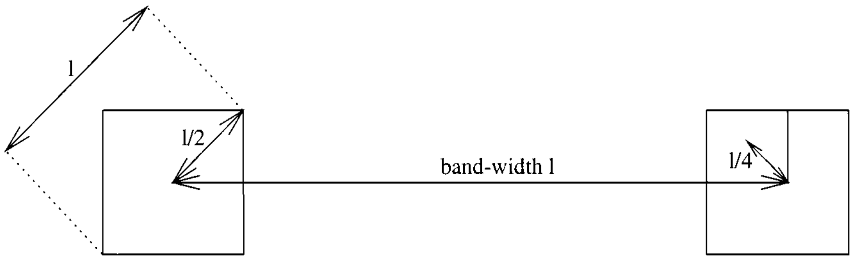

FIG. 21. Number of samples.

Minimal Size for the Clusters

For extremely large problems (over 500,000 degrees of freedom) the close interactions, which have to be computed in a traditional way and cannot be included in the FMM, may take either significant CPU time, if they are recomputed at each iteration, or significant in/out core memory, if they are stored. For example for 500,000 degrees of freedom the storage of 100 non-zero terms per line would require 1.2 GB memory.

A possible solution is to reduce the size of the clusters. However, these is a minimum size due to numerical roundoff errors. Even though the series

$$(2m+1)\, j_m(v) h_m^{(1)}(u) P_m(\cos\gamma)$$

converges for all $0 < v < u$, the sequence $h_m^{(1)}(u)$ diverges when u goes to zero. Thus with double precision computation there is a minimum distance, d, between the centers of the clusters, which is around $u \overset{\text{def}}{=} k.d \sim 1/2$ (for a relative precision of 10^{-2} for the FMM). This implies a minimal diameter for the clusters:

$$\text{Minimal diameter for the clusters} \sim \lambda/10.$$

Another constraint is the fact that the kernel for Maxwell is singular and thus analytical integration must be performed when the edges become too close. This also prevents the clusters from getting too small.

This may prove to be a serious drawback of the method when the number of triangles per wavelength increase (accumulation of points), such as around features like antennas. Greengard *et al.* [31] derived a specific formulation for the FMM to address this difficulty.

5. MEMORY MANAGEMENT

Regarding the memory management there are several possible options to reduce the memory requirements for the FMM. Here we suggest a simple algorithm which will reduce the total memory by a factor of 2.

By “tree” we denote the memory needed to construct the oct-tree with the appropriate connectivity and allocated at each node of the tree to store the radiation function sampled on S^2.

Notations:

P or Q	a cluster
L	number of levels
s_k	a point of S^2
$F_Q(s_k)$	radiation function before transfer
$G_P(s_k)$	radiation function after transfer

The radiation function $G_P(s_k)$ is constructed from $F_Q(s_k)$ with the formula

$$G_P(s_k) = G_P(s_k) + T_{l,O_P-O_Q}(s_k) F_Q(s_k).$$

The usual implementation of the FMM first allocates one “tree” for the radiation functions, $F_Q(s_k)$. Then those radiation functions are constructed starting from the smaller clusters and then moving from the lower levels to the upper levels. Once this tree is constructed, another

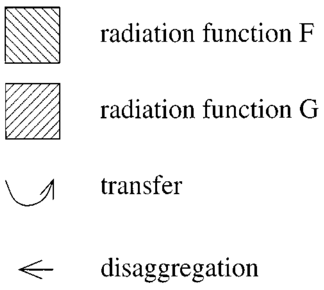

FIG. 22. Legend of Figs. 23, 24, 25, and 26.

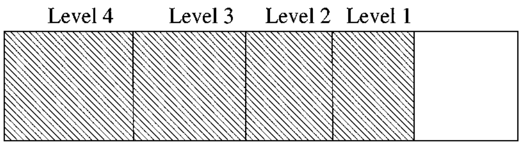

FIG. 23. First step.

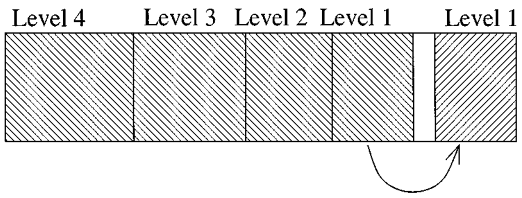

FIG. 24. Second step.

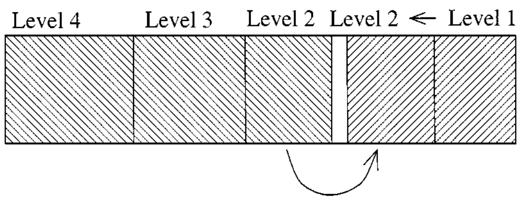

FIG. 25. Third step.

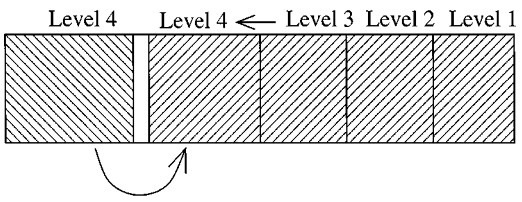

FIG. 26. Fourth step.

is allocated to store the functions $G_P(s_k)$, which are initialized by applying the transfer vector to $F_Q(s_k)$. Finally, starting from the larger clusters the information contained in $G_P(s_k)$ is propagated from the upper levels to the lower levels.

Thus with this algorithm the total amount of memory that has to be allocated is equal to two trees, one for the $F_Q(s_k)$ functions and one for $G_P(s_k)$.

We claim that it is possible to reduce this amount of memory to a minimum of one tree plus the size of the level that occupy the more memory.

First we allocate a memory space equal to one tree plus the memory used by the largest level. Then we start computing the radiation functions $F_Q(s_k)$ as before, using the first part of the memory chunk (see Fig. 23).

Then the transfers are performed for the upper level only (level 1) in the remaining chunk of memory (see Fig. 24).

The functions $F_Q(s_k)$ computed at level 1 are no longer needed. The memory space they occupied can then be reused to propagate the information from the upper level to the one below, at level 2 (disaggregation step). The transfers can then be performed at level 2 (Fig. 25) and so on until we reach the lower level (level L—Fig. 26).

6. NUMERICAL TESTS: A SPHERE

All the numerical tests are obtained on a Hewlett–Packard workstation PA-8000, with 1-GB memory.

6.1. Classical Method

This section presents the numerical results obtained with the "classical" version of the code, in which the entire matrix is assembled and stored in the core memory. The main limitation for this problem is the assembly time, which becomes prohibitive very rapidly, as well as the total memory required.

We start with test cases using the Gauss method (Table I). Here is the total CPU time, as well as the total memory used. "Size" corresponds to the size of the object in terms of wavelength. "Degrees of freedom" corresponds to the number of edges of the mesh. "Assembly" is the assembly time and "resolution" is the resolution time. "Total memory" is the memory occupied by the whole matrix.

For the last test, we used the GMRES iterative method. See Table II. GMRES converged in 19 iterations without a preconditioner, with a stopping criteria of 10^{-4}. We used the combined field integral formulation with a coefficient $\alpha = 0.2$

$$\text{CFIE} = \alpha \text{EFIE} + (1-\alpha)\frac{\imath}{\kappa}\text{MFIE}.$$

The RCS curves (Figs. 27, 28, 29, and 30) correspond to a comparison between the RCS computed numerically and the exact result computed analytically.

The quality of the results depends on the number of points per wavelength that we used, eight points per wavelength for this example, and of the error due to the fact that our mesh does not represent very accurately a sphere. For 300 edges, for example, the sphere is meshed in a rather crude way. This is why the numerical results are quite different from the analytical results.

TABLE I
Classical Method, Gauss Resolution

Size	Degrees of freedom	Assembly	Resolution	Total memory
0.67	300	7.47 s	5.02 s	1.44 MB
1	768	41.29 s	1.53 min	9.44 MB
2	3072	6.49 min	1.65 h	151 MB

TABLE II
Classical Method, Iterative Resolution

Size	Degrees of freedom	Assembly	Resolution	Total memory
2.82	5808	26 min	5.93 min	540 MB

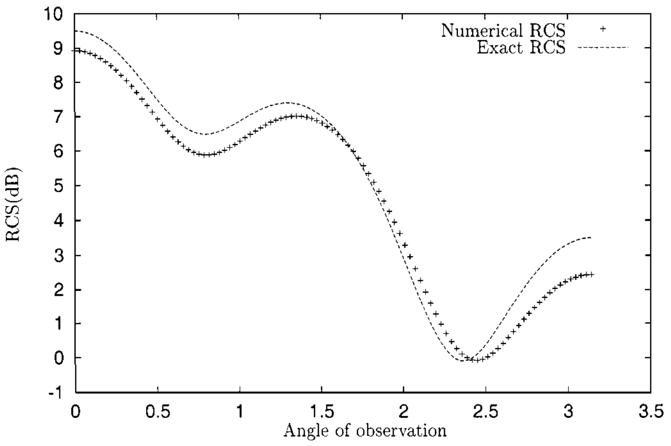

FIG. 27. 300 degrees of freedom.

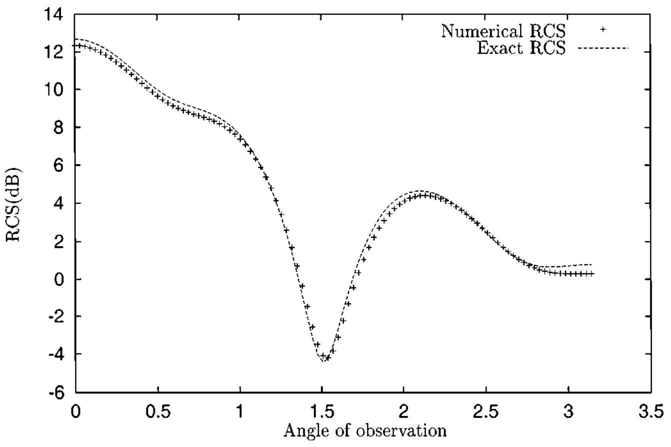

FIG. 28. 768 degrees of freedom.

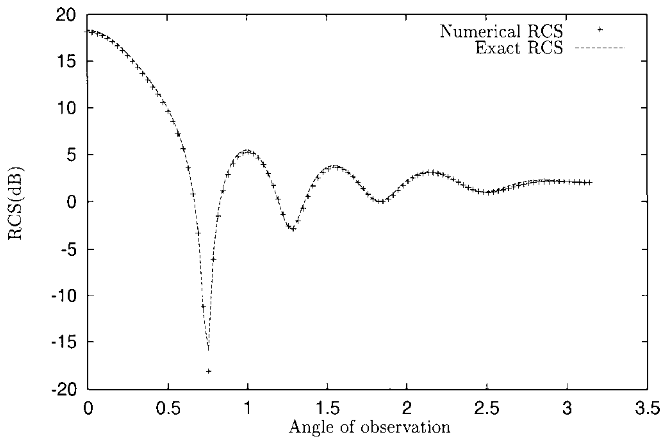

FIG. 29. 3072 degrees of freedom.

6.2. Fast Multipole Method

The results obtained with the FMM show that the FMM is very precise and that it reduces very significantly the resources needed to solve the linear system.

The formulation that we use is the combined formulation with $\alpha = 0.2$. The iterative method is GMRES (criteria equal to 10^{-4}). For this example we do not use any preconditioner.

We begin with the same tests as with the classical method. “Far away interactions” corresponds to the memory space occupied by the FMM. “Close interactions” corresponds to the central band of the matrix, i.e., the part of matrix which has to be computed exactly with the classical method. “Assembly” corresponds to the CPU time needed to compute the central band. The rest of the assembly time, for the FMM, is negligible. “Iterations” is the number of iterations needed for GMRES to converge. See Tables III and IV.

The RCS (Figs. 31, 32, 33, and 34) is completely identical to the one computed before.

We now present the relative difference between the FMM and the classical method with two different norms: l^∞ and l^2. This error is stable and independent of the size of the problem. See Table V.

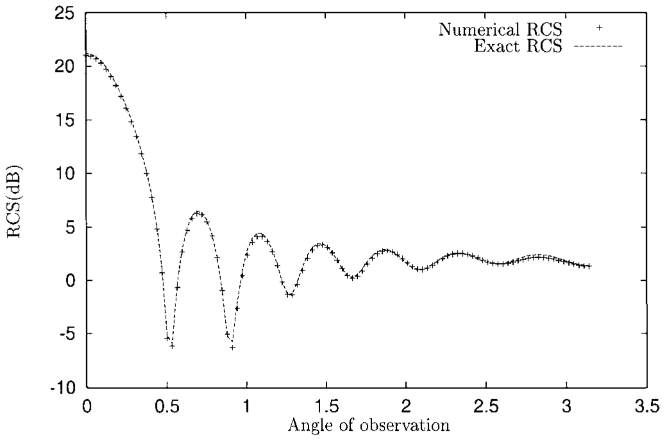

FIG. 30. 5808 degrees of freedom.

TABLE III
FMM for the Sphere, CPU Time

Size	Degrees of freedom	Assembly	Resolution	Iterations
0.67	300	3.6 s	14.1 s	13
1	768	9.83 s	37.6 s	15
2	3072	41.3 s	3.68 min	18
2.82	5808	1.3 min	8.7 min	19

TABLE IV
FMM for the Sphere, Memory Space

Size	Degrees of freedom	Far-away interactions	Close interactions
0.67	300	0.31 MB	0.47 MB
1	768	0.58 MB	1.4 MB
2	3072	2.7 MB	5.4 MB
2.82	5808	5.7 MB	10 MB

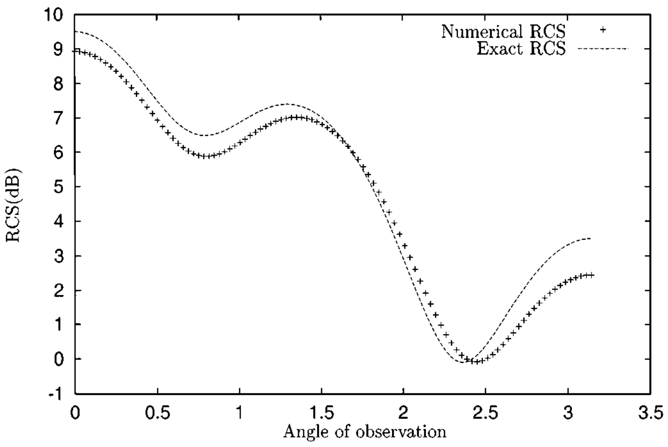

FIG. 31. 300 degrees of freedom.

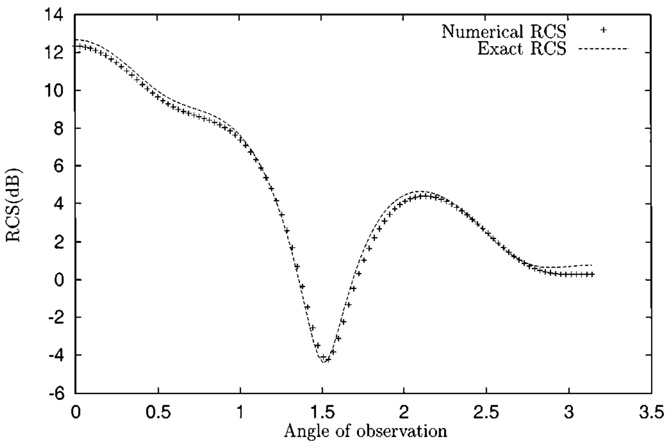

FIG. 32. 768 degrees of freedom.

TABLE V
Relative Error for the FMM

Size	Degrees of freedom	Norm l^∞	Norm l^2
0.67	300	0.00413	0.00273
1	768	0.00681	0.00288
2	3072	0.00917	0.00386
2.82	5808	0.00728	0.00393

TABLE VI
FMM for the Sphere, Small Wavelength, CPU Time

Size	Degrees of freedom	Assembly	Resolution	Iterations
4	11,532	2.83 min	20.4 min	21
8	47,628	14.1 min	1.5 h	25
11.3	95,052	38.7 min	3.7 h	28

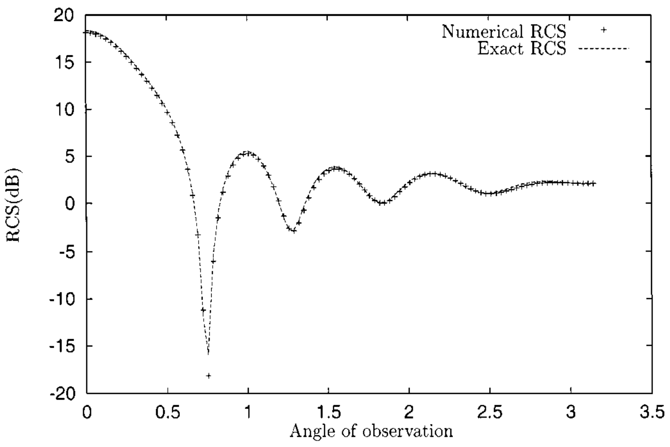

FIG. 33. 3072 degrees of freedom.

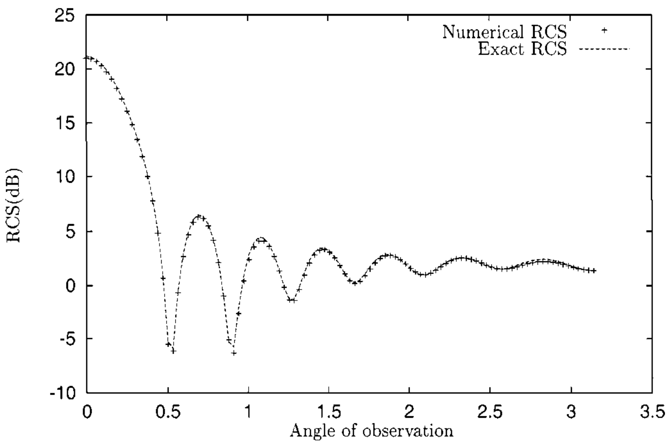

FIG. 34. 5808 degrees of freedom.

TABLE VII
FMM for the Sphere, Small Wavelength, Memory Space

Size	Degrees of freedom	Far-away interactions	Close interactions
4	11,532	11.4 MB	19.5 MB
8	47,628	48.4 MB	83.1 MB
11.3	95,052	102 MB	165 MB

Thanks to the FMM, we are able to solve much larger problems. Here are the results obtained when increasing progressively the wavelength. See Tables VI and VII.

The RCS curves are shown in Figs. 35, 36, and 37.

A zoom in the shadow region, for the case with 95,052 degrees of freedom, shows that the solution oscillates around the exact value: see Fig. 38.

Those oscillations are due to the errors in the numerical resolution. We observe that the RCS reaches a maximum of 2000. The exact value for $\theta = \pi$ (shadow region) is equal to 1.55. The difference between the two is very large. Thus the resolution errors become apparent. The difference between the RCS for $\theta = 0$ and $\theta = \pi$ increases with the frequency. Thus we expect that this oscillation phenomena will increase with the frequency.

To illustrate this observation (Fig. 39), here is the result obtained by simply increasing the number of points per wavelength:

$$\text{Number of points per wavelength:} \qquad 8 \rightarrow 11.3.$$

Those oscillations are reduced by a factor of 1.7. The oscillations around the exact value, with 8 points per wavelength, are equal to 0.44 for θ between 3 and π (be careful: In Fig. 38 the RCS is shown in decibels). This difference is reduced to 0.26 with 11.3 points per wavelength.

6.3. Convergence of the FMM

The number of terms used in the series (13), Section 4, is computed with the formula

$$l = v + C\log(\pi + v).$$

See Eq. (14).

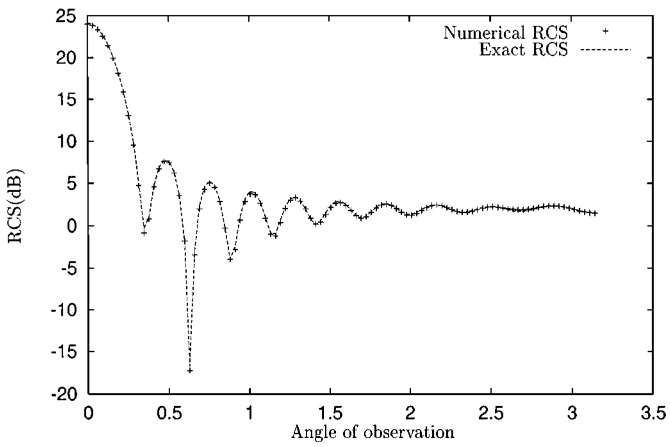

FIG. 35. 11,532 degrees of freedom.

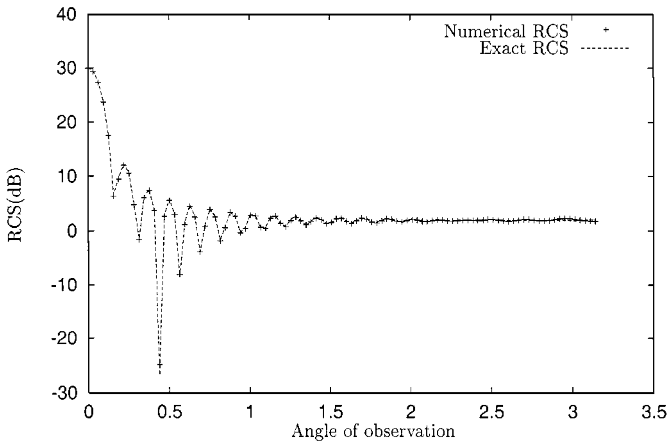

FIG. 36. 47,628 degrees of freedom.

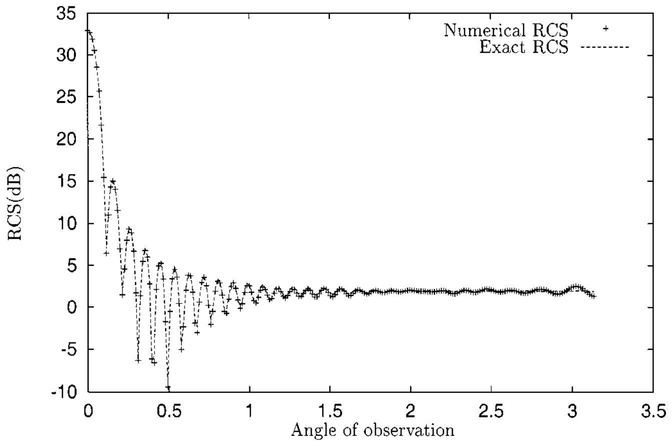

FIG. 37. 95,052 degrees of freedom.

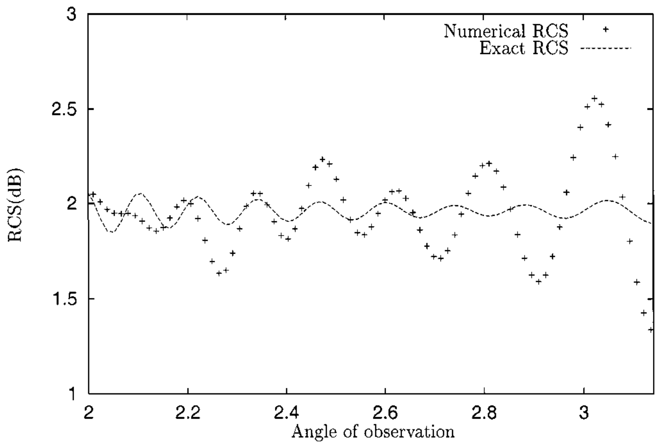

FIG. 38. 95,052 degrees of freedom: zoom.

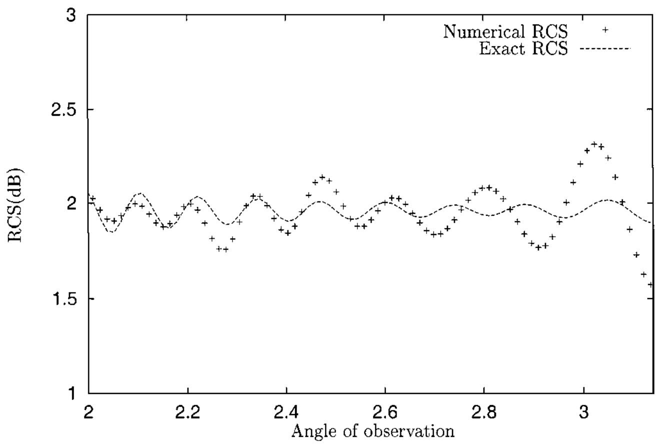

FIG. 39. 95,052 degrees of freedom: zoom, 11.3 points per wavelength.

We choose different values for C ranging from 1 to 6, in order to observe the reduction of the error due to the FMM. The value used for the previous tests was 2.25. This is a good compromise between the CPU time and the numerical precision. See Table VIII.

The RCS curves are all identical to the ones obtained with the classical method. The difference is too small to be observed.

We observe that after 4 the convergence is stopped. This is due to the numerical instabilities of the method. Let us describe more precisely the cause of those instabilities.

Numerical Instabilities

The numerical error for the FMM depends on the convergence of the sequence

$$(2m+1)\, j_m(v) h_m^{(1)}(u) P_m(w).$$

The numerical instabilities are linked to the asymptotic behavior of the Bessel functions, which we recall: for $m \to +\infty$,

$$j_m(v) \sim \frac{v^m}{(2m+1)(2m-1)\cdots 3\cdot 1}$$

$$y_m(u) \sim -\frac{(2m-1)(2m-3)\cdots 3\cdot 1}{u^{m+1}}.$$

TABLE VIII
Relative Error as a Function of the Number of Poles in the Expansion

Value of C	Norm l^{∞}	Norm l^2	CPU time
1	0.044	0.0189	4.32 min
2	0.015	0.0069	7.97 min
3	0.0068	0.0032	12.8 min
4	0.00689	0.00304	27.7 min
5	0.00696	0.00303	30.9 min
6	0.00689	0.00302	35.4 min

The product $j_m(v)h_m^{(1)}(u)$ converges to 0 but $h_m^{(1)}(u) \to +\infty$ when $m \to +\infty$. The divergence occurs when $m \geq u$. For a stable implementation, it is thus necessary to guarantee that m is inferior to u.

Let us consider the smallest clusters, at the lowest level. We denote by a the length of an edge of one of those clusters.

The factor v corresponds to the size of a cluster ($=\sqrt{3}\kappa a$), while the factor u corresponds to the transfer vector, the norm of which is equal to the distance between the centers of two clusters times κ. The minimum value for u is thus equal to $2\kappa a$. We have $u_{\min} = \frac{2}{\sqrt{3}}v$.

Since the number of terms in the series is given by

$$l = v + C\log(\pi + v),$$

we observe a divergence if $v + C\log(\pi + v) \geq u$. Thus the stability condition reads

$$v + C\log(\pi + v) < v\frac{2}{\sqrt{3}} \tag{15}$$

$$\Leftrightarrow C < v\frac{\frac{2}{\sqrt{3}} - 1}{\log(\pi + v)} \sim 0.15\frac{v}{\log(\pi + v)}. \tag{16}$$

When v tends to infinity this condition is less and less restrictive since $v/\log(\pi + v) \to +\infty$. However, instabilities might appear if v is too small.

As a conclusion, we have to choose C depending on the desired accuracy and then a minimal size for the clusters, v, such that the stability condition is satisfied.

In the code, the length of the side of one cluster is equal to

$$a = \frac{D}{\kappa},$$

where, for our example, $D = 1$. For our test case, $a = 0.11$ and the average length of an edge is equal to 0.089. The average bandwidth for the central band (close interactions) is equal to 86. It is fairly small but for 100,000 edges the storage is already 206 MB. This is why we tried to reduce it as much as we could.

To illustrate this divergence, we give the maximum value of the transfer function on S^2, depending on C and D. See Table IX.

This confirms our analysis of the stability.

Those problems may become a serious obstruction if there is a large concentration of points in a small area of the scattering object. This leads to an increase in the part of the matrix corresponding to the close interactions. We refer the reader to a recent article by Greengard *et al.* [31] that addresses this problem.

TABLE IX
Divergence of the Transfer Function

Coefficient C	$D = 1$	$D = 2$	$D = 4$
1	4	4	4
2	10	7	4
3	10^2	50	50
4	3.10^3	2.10^3	4.10^2
5	3.10^5	3.10^5	2.10^5
6	10^6	7.10^5	2.10^5

7. ASYMPTOTIC GROWTH OF THE COMPLEXITY

The scattering from a sphere allows us to test the FMM on a large band of frequency and to observe the growth of the CPU time and memory.

Academic Case

The first numerical test corresponds to a matrix–vector product Mu, where the matrix M is defined by

$$M_{ij} = \begin{cases} \frac{e^{\iota\kappa|x_i - x_j|}}{|x_i - x_j|} & \text{if } i \neq j \\ 0 & \text{otherwise} \end{cases}.$$

Each coordinate u_i of u is a random float number. The points x_i are uniformly distributed on the surface, with 10 points per wavelength. The radius of the sphere is equal to one.

We reach 1,000,000 points on the surface.

The CPU time corresponds to a single matrix–vector product. The complexity curve follows very well the theoretical predicted curve $O(N \log^2 N)$. See Fig. 40.

Scattering from a Sphere

We now measure the CPU time of a single matrix–vector product where this time the matrix is the CFIE matrix

$$\text{CFIE} = \alpha\text{EFIE} + (1-\alpha)\frac{\iota}{\kappa}\text{MFIE}.$$

See Figs. 41 and 42.

The test cases correspond to those presented in Section 6.2. However in Fig. 41, the CPU time for *one matrix–vector product* only is represented. In Section 6.2, the CPU time corresponds to the complete resolution of the linear system with an iterative method.

The curves (Figs. 41 and 42) show that the CPU time and the memory space behave asymptotically like the theory predicts. Moreover, they show that the FMM should not be used if the scattering object is too small. Typically the FMM becomes efficient as soon as the size of the object is superior to a few wavelengths.

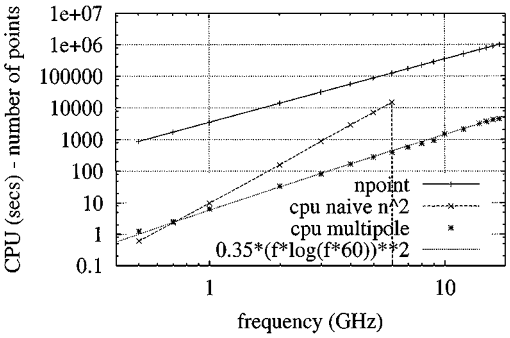

FIG. 40. Academic case: matrix–vector product with the FMM.

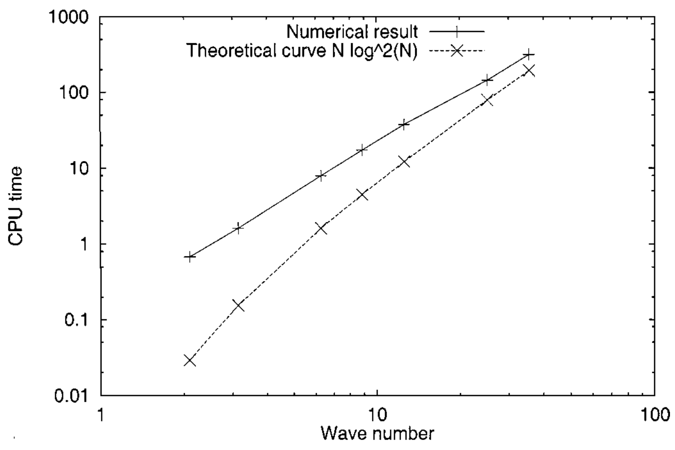

FIG. 41. CPU time for a product with the CFIE matrix.

8. NUMERICAL TEST: AN INDUSTRIAL APPLICATION

We now study an industrial test case, the CETAF. This is a standard test case which allows comparison with available codes from aerospace companies. There is no analytical solution to this problem. The object is shown in Fig. 43.

We computed the currents induced by the incident wave for the frequencies 7 and 15 GHz.

8.1. 7-GHz Case

The mesh has 45,960 edges. Its size is 11.7 wavelengths. We have 9.84 edges per wavelength on average.

The incident wave hits the CETAF with an angle of 45°.

wavelength	0.0428
shortest edge	0.00625
longest edge	0.000674

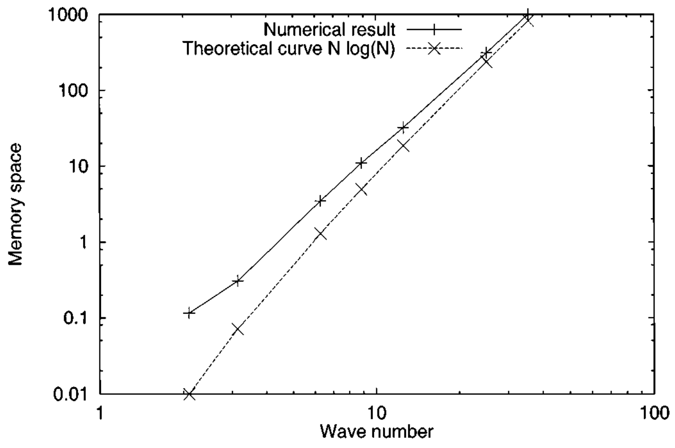

FIG. 42. Memory space used by the FMM.

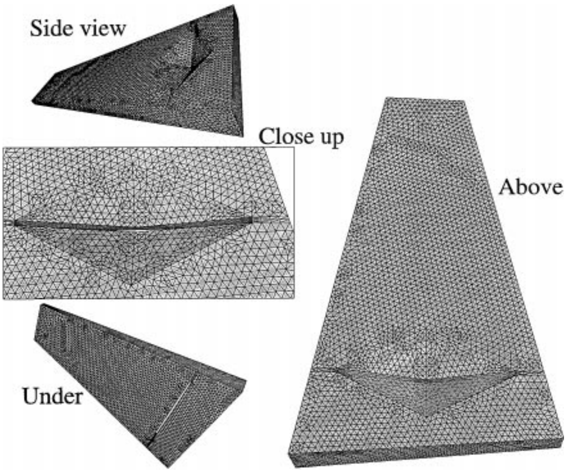

FIG. 43. CETAF.

The performance are shown in Table X. The linear system was solved with GMRES and was preconditioned with SPAI, which produces a sparse approximate inverse. See [32] for a description of SPAI and [4] for an application of SPAI to the resolution of integral equations for Maxwell; see also [6, 20].

Results

A comparison of the RCS computed with our code and with Dassault-Aviation SPECTRE code is given in Fig. 44. Dassaut-Aviation SPECTRE uses a classical method with a full assembly of the matrix and the linear system was solved on a parallel computer with a Gauss method.

This curve shows two peaks which can be interpreted as follows. The first peak occurs for $\theta = 45^\circ + 90^\circ = 135^\circ$. This corresponds to a reflection of the incident wave on the surface of the CETAF. The second peak corresponds to the shadow region. In this region the scattered field cancels the incident field. Thus it is a region in which the radiated energy is very important. This occurs for $\theta = 45^\circ + 180^\circ = 225^\circ$. These peaks are very classical in RCS problems.

We also present a comparison for the current on the surface of the CETAF. The intensity of the current is shown in Figs. 46 and 47 for the shadow region. Finally the y-coordinate of the real part of the current is shown in Fig. 48. We observe stripes with a period equal to the period of the incident wave.

TABLE X
Performances for the 7-GHz Case

Size:	11.7λ
Degrees of freedom:	45,960
Assembly:	12.8 min
Resolution:	2.58 h
Iterations:	47
Far-away interactions:	31.7 MB
Close interactions:	110 MB

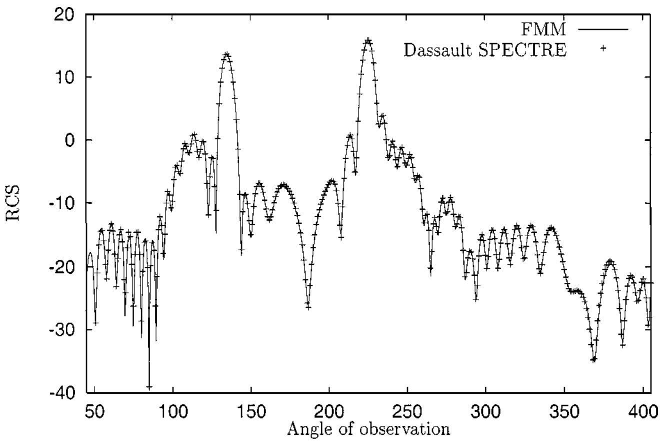

FIG. 44. 45,000 degrees of freedom, 7 GHz, RCS.

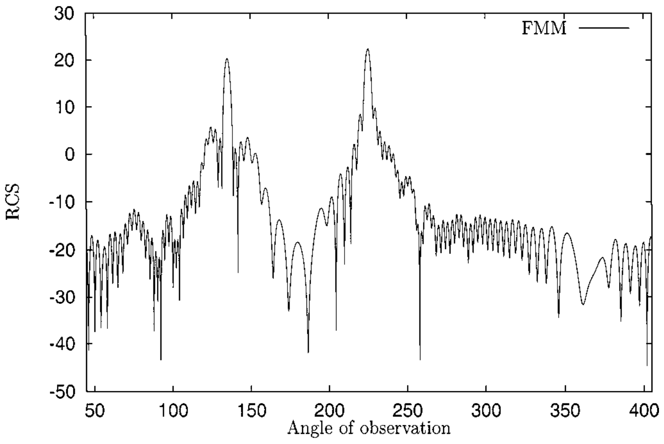

FIG. 45. 200,000 degrees of freedom, 15 GHz, RCS.

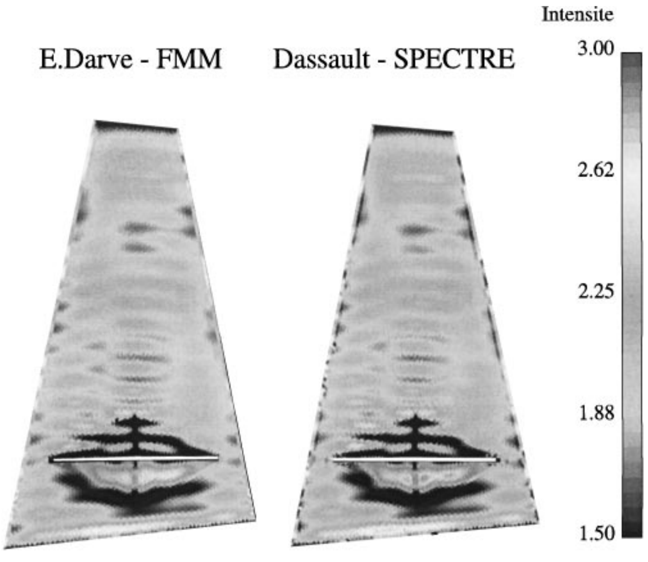

FIG. 46. 45,000 degrees of freedom, 7 GHz: above.

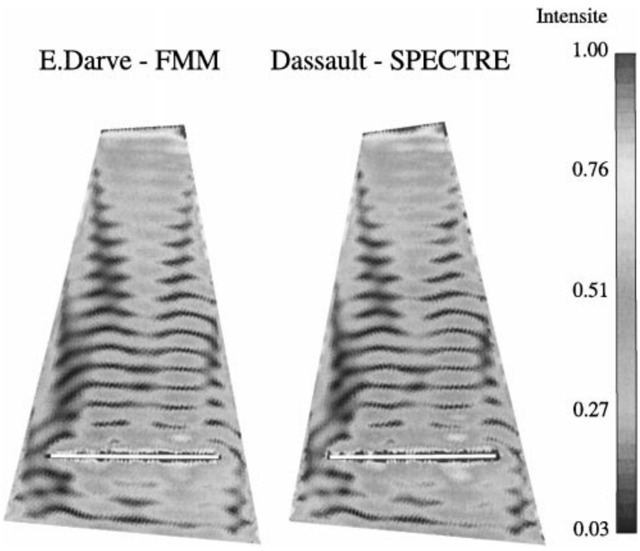

FIG. 47. 45,000 degrees of freedom, 7 GHz: shadow region.

The FMM code uses a combined formulation magnetic + electric. To the contrary Dassault-Aviation SPECTRE uses an EFIE formulation. This is the main reason that we observe a slight difference between the two solutions. The error introduced by the FMM is much smaller than the difference between the two formulations. These tests mostly illustrate the fact that EFIE and CFIE are equivalent when the size of the edges goes to zero but they do not produce exactly the same solution if the number of points per wavelength is not sufficient.

8.2. 15-GHz Case

The RCS is represented in Fig. 45. This case was too large to be solved with SPECTRE. Thus we do not present any comparison this time.

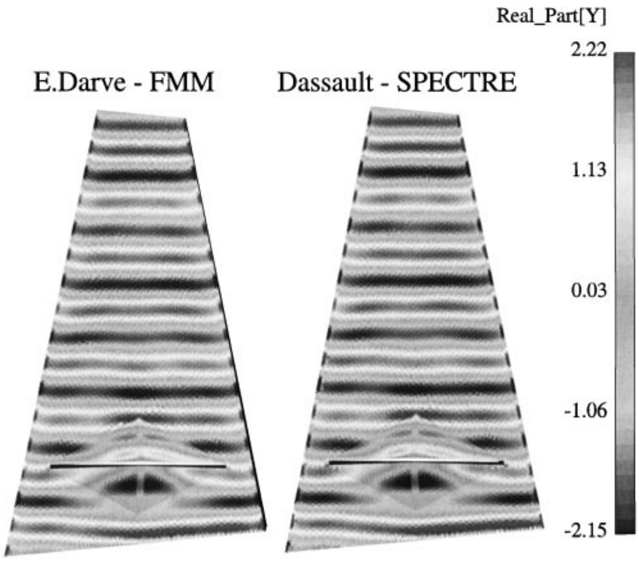

FIG. 48. 45,000 degrees of freedom, 7 GHz: y-coordinate of the real part.

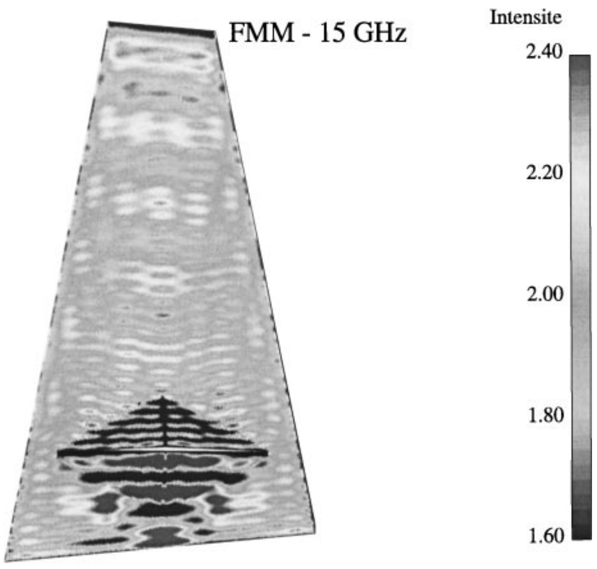

FIG. 49. 200,000 degrees of freedom, 15 GHz: above.

The intensity of the current is shown in Figs. 49 and 50 for the shadow region. The performances are given in Table XI.

We also gave new figures, the FMM CPU time and the “close interactions CPU time,” because for this case we had to adapt the implementation. For such a matrix, the central band (close interactions) is too large to be stored. Thus the entries in the central band are either recomputed if the double integral on the surface is numerical (small recomputation time) or stored if the double integral has to be computed analytically (significant CPU time). Thus only the “singular” integrals are stored. This is why the assembly time and storage for the close interactions are relatively small. Moreover, since at each iteration some entries

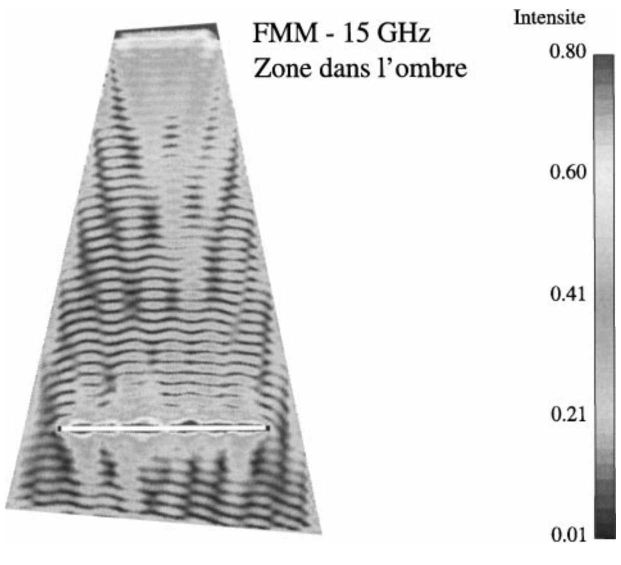

FIG. 50. 200,000 degrees of freedom, 15 GHz: shadow region.

TABLE XI
Performances for the 15-GHz Case

Wavelength:	0.02
Number of points per wavelength:	9.18
Size:	25.0λ
Degrees of freedom:	183,840
Assembly:	24.4 min
Resolution:	19.7 h
FMM CPU time:	13.3 h
Close interactions CPU time:	6.5 h
Iterations:	53
Far-away interactions:	140 MB
Close interactions:	170 MB

have to be recomputed, we separated the recomputation time for those entries (6.5 h total) from the FMM CPU time (13.3 h total). See table XI.

9. CONCLUSION

The FMM reduces the complexity of the resolution of the dense matrix from N^3 (inversion with the Gauss method) to a minimum of $N \log N$. In this article are reviewed the difficulties appearing in the implementation of the method. Improving the implementation results in a larger domain of application of the FMM, which becomes competitive, compared to a naive implementation with a dense matrix, at even lower N. A particularly crucial point of the implementation is the interpolation and anterpolation steps. We here presented a variant of the scheme proposed by Chew and Lu in [41]. Its complexity is $O(K)$, if K is the number of samples in S^2. Compared to Chew and Lu's algorithm, our scheme improves the accuracy while reducing the CPU time. This scheme has to be seen as an alternative to the semi-naive scheme with complexity $K^{1.5}$, which performs better for the small clusters. There are other schemes, albeit with higher asymptotic complexity ($K \log K$), proposed by Alpert and Jakob-Chien and Dembart and Yip (see [5, 24]), to which we have not compared but it is estimated that the new scheme should be used for problems larger than 50,000 unknowns.

Finally we would like to point out that a significant problem was not addressed in this paper, which is the parallelization of the algorithm. This is a very active area. We will refer the reader to some very recent publications on this topic: [1, 2, 7, 8, 33, 37, 51, 52].

ACKNOWLEDGMENTS

I thank G. Alléon, Aérospatiale Espace et Défense; M. Mallet and F. Dubois, Dassault-Aviation; and O. Pironneau, Pierre et Marie Curie University for their contribution to this work.

REFERENCES

1. A. Grama, V. Kumar, and A. Sameh, Parallel hierarchical solvers and preconditioners for boundary-element methods, *SIAM J. Sci. Comput.* **20**(1), 337 (1999).
2. A. Nakano, Parallel multilevel preconditioned conjugate-gradient approach to variable-charge molecular-dynamics, *Comput. Phys. Commun.* **104**(1–3), 59 (1997).

3. A. H. Boschitsch, M. O. Fenley, and W. K. Olson, A fast adaptive multipole algorithm for calculating screened coulomb (yukawa) interactions, *J. Comput. Phys.* **151**, 212 (1999).
4. G. Alléon, M. Benzi, and L. Giraud, *Sparse Approximate Inverse Preconditioning for Dense Linear Systems Arising in Computational Electromagnetics*, Technical Report TR/PA/97/05, CERFACS, Toulouse, France, April 1997.
5. B. Alpert and R. Jakob-Chien, A fast spherical filter with uniform resolution, *J. Comput. Phys.* **136**, 580 (1997).
6. M. Benzi and M. Tuma, A sparse approximate inverse preconditioner for nonsymmetric linear systems, *SIAM J. Sci. Comput.* **19**(3), 968 (1998).
7. B. G. Johnson, The continuous fast multipole method: Large-scale density-functional calculations on parallel computing platforms, *Abs. Papers Am. Chem. Soc.* **213**(1), 272 (1997).
8. B. G. Johnson and R. L. Graham, The parallel continuous fast multipole method, *Abs. Papers Am. Chem. Soc.* **213**(2), 25 (1997).
9. S. S. Bindiganavale and J. L. Volakis, Comparison of three fmm techniques for solving hybrid fe-bi systems, *IEEE Antennas Propag. Mag.* **39**(4), 47 (1997).
10. O. M. Bucci and G. Franceschetti, On the degrees of freedom of scattered fields, *IEEE Trans. Antennas Propag.* **37**, 918 (1989).
11. O. M. Bucci, C. Gennarelli, R. Riccio, and C. Savarese, Electromagnetic fields interpolation from nonuniform samples over spherical and cylindrical surfaces, *IEEE Proc. Microwave Antennas Propag.* **141**, 77 (1994).
12. O. M. Bucci, C. Gennarelli, and C. Savarese, Optimal interpolation of radiated fields over a sphere, *IEEE Trans. Antennas Propag.* **AP-39**, 1633 (1991).
13. C. Ochsenfeld, C. A. White, and M. Headgordon, Linear and sublinear scaling formation of Hartree–Fock type exchange matrices, *J. Chem. Phys.* **109**, 1663 (1998).
14. C. A. White, B. G. Johnson, P. M. W. Gill, and M. Headgordon, A generalized fast multipole approach for Hartree-Fock and density-functional computations: Comment, *Chem. Phys. Lett.* **248**(5–6), 482 (1996).
15. C. A. White, B. G. Johnson, P. M. W. Gill, and M. Headgordon, Linear scaling density-functional calculations via the continuous fast multipole method, *Chem. Phys. Lett.* **253**(3–4), 268 (1996).
16. C. A. White and M. Headgordon, Derivation and efficient implementation of the fast multipole method, *J. Chem. Phys.* **101**(8), 6593 (1994).
17. C. A. White and M. Headgordon, Fractional tiers in fast multipole method calculations, *Chem. Phys. Lett.* **257**, 647 (1996).
18. C. A. White and M. Headgordon, Rotating around the quartic angular-momentum barrier in fast multipole method calculations, *J. Chem. Phys.* **105**(12), 5061 (1996).
19. W. C. Chew, Y. M. Wang, and L. Gürel, Recursive algorithm for wave-scattering solutions using windowed addition theorem, *J. Electromag. Waves Appl.* **6**(11), 1537 (1992).
20. E. Chow and Y. Saad, Approximate invese preconditioners via sparse–sparse iterations, *SIAM J. Sci. Comput.* **19**(3), 995 (1998).
21. R. Coifman, V. Rokhlin, and S. Wandzura, The fast multipole method for the wave equation: A pedestrian prescription, *IEEE Antennas Propag. Mag.* **35**(3), 7 (1993).
22. E. Darve, The fast multipole method (i): Error analysis and asymptotic complexity, *SIAM Numer. Anal.*, to appear.
23. B. Dembart and G. R. Shubin, *A 3d fast Multipole Method for Electromagnetics with Multiple Levels*, Technical document ISSTECH-97-004, Boeing, December 1994.
24. B. Dembart and E. Yip, A 3d fast multipole method for electromagnetics with multiple levels, in *Proceedings of the 11th Annual Review of Progress in Applied Computational Electromagnetics, Monterey, CA, March 1995*, Vol. 1, p. 621.
25. A. Dutt, M. Gu, and V. Rokhlin, Fast algorithms for polynomial interpolation, integration and differentiation, *SIAM J. Numer. Anal.* **33**(5), 1996.
26. N. Engheta, W. D. Murphy, V. Rokhlin, and M. S. Vassiliou, The fast multipole method (fmm) for electromagnetic scattering problems, *IEEE Trans. Antennas Propag.* **40**(6), 634 (1992).
27. E. R. Smith, Fast moment methods for general potentials in molecular dynamics, *Mol. Phys.* **86**(4), 769 (1995).

28. G. T. D. Francia, Directivity, super-gain and information, *IRE Trans. Antennas Propag.* **4**, 473 (1956).

29. G. T. D. Francia, Degrees of freedom of an image, *J. Opt. Soc. Am.* **59**, 779 (1969).

30. L. Greengard, Fast algorithms for classical physics, *Science* **265**(5174), 909 (1994).

31. L. Greengard, J. Huang, V. Rokhlin, and S. Wandzura, Accelerating fast multipole methods for low frequency scattering, *IEEE Comput. Sci. Eng. Mag.* (July 1998).

32. M. J. Grote and T. Huckle, Parallel preconditioning with sparse approximate inverses, *SIAM J. Sci. Comput.* **18**(3), 838 (1997).

33. H. Schwichtenberg, G. Winter, and H. Wallmeier, Acceleration of molecular mechanic simulation by parallelization and fast multipole techniques, *Parallel Comput.* **25**(5), 535 (1999).

34. H. G. Petersen, D. Soelvason, and J. W. Perram, The very fast multipole method, *J. Chem. Phys.* **101**(10), 8870 (1994).

35. H. G. Petersen, D. Soelvason, J. W. Perram, and E. R. Smith, Error-estimates for the fast multipole method. 1. The 2 dimensional case, *Proc. Roy. Soc. London Ser. A* **448**(1934), 389 (1995).

36. H. G. Petersen, E. R. Smith, and D. Soelvason, Error-estimates for the fast multipole method. 2. The 3 dimensional case, *Proc. Roy. Soc. London Ser. A* **448**(1934), 401 (1995).

37. J. A. Board, Introduction to "a fast algorithm for particle simulations," *J. Comput. Phys.* **135**, 279 (1997).

38. J. Y. Lee and K. Jeong, A parallel poisson solver using the fast multipole method on networks of workstations, *Comput. Math. Appl.* **36**(4), 47 (1998).

39. J. J. Knab, Interpolation of band-limited functions using the approximate prolate series, *IEEE Trans. Inform. Theory*, 717 (1979).

40. J. J. Knab, The sampling window, *IEEE Trans. Inform. Theory*, 157 (1983).

41. S. Koc, J. M. Song, and W. C. Chew, *Error Analysis for the Numerical Evaluation of the Diagonal Forms of the Scalar Spherical Addition Theorem*, Research Report CCEM-32-97, University of Illinois, October 1997.

42. L. Greengard and V. Rokhlin, A fast algorithm for particle simulations, *J. Comput. Phys.* **135**, 280 (1997); reprinted from *J. Comput. Phys.* **73**, 325 (1987).

43. C. C. Lu and W. C. Chew, A multilevel algorithm for solving a boundary integral equation of wave scattering, *Micro. Opt. Tech. Lett.* **7**(10), 466 (1994).

44. M. Challacombe, C. White, and M. Headgordon, Periodic boundary conditions and the fast multipole method, *J. Chem. Phys.* **107**(23), 10131 (1997).

45. M. Challacombe, E. Schwegler, C. White, B. Johnson, P. Gill, and M. Headgordon, Linear scaling computation of hf and hf/dft fock matrices. *Abs. Papers Am. Chem. Soc.* **213**(2), 57 (1997).

46. A. D. McLaren, Optimal numerical integration on a sphere, *Math. Comput.* **17**, 361 (1963).

47. E. Michielssen and A. Boag, Multilevel evaluation of electromagnetic fields for the rapid solution of scattering problems, *Microwave Opt. Tech. Lett.* **7**(17), 790 (1994).

48. E. Michielssen and A. Boag, A multilevel matrix decomposition algorithm for analyzing scattering from large structures, in *11th Annual Review of Progress in Applied Computational Electromagnetics, Monterey, CA, 1995*, p. 614.

49. S. M. Rao, D. R. Wilton, and A. W. Glisson, Electromagnetic scattering by surfaces of arbitrary shape, *IEEE Trans. Antennas Propag.* **AP-30**(3), 409 (1982).

50. V. Rokhlin, Rapid solution of integral equations of scattering theory in two dimensions, *J. Comput. Phys.* **86**, 414 (1990).

51. V. Rokhlin, *Diagonal Forms of Translation Operators for the Helmholtz Equation in Three Dimensions*, Research Report YALEU/DCS/RR-894, Yale University Department of Computer Science, March 1992.

52. V. Rokhlin, Analysis-based fast numerical algorithms of applied mathematics, in *Proceedings of the International Congress of Mathematicians, Zürich, Switzerland, 1994.*

53. S. H. Teng, Provably good partitioning and load balancing algorithms for a parallel adaptive n-body simulation, *SIAM J. Sci. Comput.* **19**(2), 635 (1998).

54. S. K. Goh, C. P. Sosa, and A. Stamant, A Scalable divide-and-conquer algorithm combining coarse and fine-grain parallelization, *Theor. Chem. Accounts* **99**(3), 197 (1998).

55. J. M. Song and W. C. Chew, Multilevel fast-multipole algorithm for solving combined field integral equations of electromagnetic scattering, *Microwave Opt. Tech. Lett.* **10**(1), 14 (1995).

56. J. M. Song, C.-C. Lu, and W. C. Chew, Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects, *IEEE Trans. Antennas Propag.* **45**(10), 1488 (1997).

57. J. M. Song, C.-C. Lu, W. C. Chew, and S. W. Lee, *Fast Illinois Solver Code (Fisc) Solves Problems of Unprecendented Size at the Center for Computational Electromagnetics, University of Illinois*, Technical Report CCEM-23-97, University of Illinois, Urbana Champaign, 1997.

58. J. M. Song, C. C. Lu, W. C. Chew, and S. W. Lee, Fast Illinois solver code (fisc), *IEEE Antennas Propag Mag.* **40**(3), 27 (1998).

59. T. Schlick, R. D. Skeel, A. T. Brunger, L. V. Kalé, J. A. Board, J. Hermans, and K. Schulten, Algorithmic challenges in computational molecular biophysics, *J. Comput. Phys.* **151**, 9 (1999).

60. R. L. Wagner and W. C. Chew, A ray-progagation fast multipole algorithm, *Microwave Opt. Tech. Lett.* **7**(10), 435 (1994).

61. W. D. Elliott and J. A. Board, Fast Fourier-transform accelerated fast multipole algorithm, *SIAM J. Sci. Comput.* **17**(2), 398 (1996).

62. C. A. White, B. G. Johnson, P. M. W. Gill, and M. Head-Gordon, The continuous fast multipole method, *Chem. Phys. Lett.* **230**(1–2), 8 (1994).

63. N. Yarvin and V. Rokhlin, An improved fast multipole algorithm for potential fields on the line, *SIAM J. Numer. Anal.* **36**(2), 629 (1999).

64. N. Yarvin and V. Rokhlin, A generalized one-dimensional fast multipole method with application to filtering of spherical harmonics, *J. Comput. Phys.* **147**, 594 (1998).