A Well-Scaling Parallel Algorithm for the Computation of the Translation Operator in the MLFMA

Bart Michiels, Ignace Bogaert, *Member, IEEE,* Jan Fostier, *Member, IEEE,* and Daniël De Zutter, *Fellow, IEEE*

***Abstract*—This paper investigates the parallel, distributed-memory computation of the translation operator with $L+1$ multipoles in the three-dimensional Multilevel Fast Multipole Algorithm (MLFMA). A baseline, communication-free parallel algorithm can compute such a translation operator in $\mathcal{O}(L)$ time, using $\mathcal{O}(L^2)$ processes. We propose a parallel algorithm that reduces this complexity to $\mathcal{O}(\log L)$ time. This complexity is theoretically supported and experimentally validated up to 16 384 parallel processes. For realistic cases, the implementation of the proposed algorithm proves to be up to ten times faster than the baseline algorithm. For a large-scale parallel MLFMA simulation with 4096 parallel processes, the runtime for the computation of all translation operators during the setup stage is reduced from roughly one hour to only a few minutes.**

***Index Terms*—MLFMA, translation operator, parallel computing, distributed-memory architecture**

I. Introduction

Electromagnetic scattering problems involving piecewise homogeneous objects are often formulated using boundary integral equations. A Method of Moments (MoM) discretization then yields a dense set of N linear equations and N unknowns. When solving this set of equations iteratively, the Multilevel Fast Multipole Algorithm (MLFMA) can be used to evaluate the matrix-vector multiplication with a complexity of only $\mathcal{O}(N\log N)$ [1]–[3], allowing the solution for large problems. Within the MLFMA, unknowns are hierarchically organized into an octree of boxes and interactions between these boxes are evaluated using radiation patterns and translation operators. In the three-dimensional MLFMA, the translation operator with $L+1$ multipoles is given by [2], [3]

$$T(\vec{k},\vec{R}_T)=\sum_{l=0}^{L}(-j)^l(2l+1)h_l^{(2)}(kR_T)P_l(\cos\theta_T) \tag{1}$$

with $\cos\theta_T=\vec{1}_k\cdot\vec{1}_{R_T}$, $\vec{k}=k\vec{1}_k$ a vector representing the angular direction in which the translation operator is to be evaluated, k the wavenumber, $\vec{R}_T=R_T\vec{1}_{R_T}$ the translation direction connecting the centers of the two interacting boxes and $P_l(.)$ and $h_l^{(2)}(.)$ the Legendre polynomial and spherical Hankel function of the second kind of order l respectively. Given a fixed k and $\vec{R}_T$, the translation operator T is a one-dimensional function of θ_T.

The authors are with the Department of Information Technology (INTEC), Ghent University, Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium (e-mail: bart.michiels@intec.ugent.be).

Manuscript received August 6, 2013; revised January 19, 2014.

In the MLFMA, translation operators with $L+1$ multipoles are sampled in $\mathcal{O}(L^2)$ angular points. The direct calculation of a translation operator using (1) hence requires $\mathcal{O}(L^3)$ operations. We refer to this method as the "direct method" (DM). In [4], a two-step procedure was introduced to reduce this complexity to $\mathcal{O}(L^2)$ [4], [5]. In the first step, the bandlimited function $T(\theta_T)$ is evaluated in $\mathcal{O}(L)$ equidistant points in the θ_T-dimension, ranging from 0 to π, using (1). In the second step, local interpolation is used to evaluate T in the required $\mathcal{O}(L^2)$ points. Both steps require $\mathcal{O}(L^2)$ time. This method is referred to as the "interpolation method" (IM).

In this paper, we investigate an algorithm for the parallel, distributed-memory computation of the translation operator. Even though this problem is interesting in its own right, the main motivation for our work is closely related to recent advances in the development of distributed-memory, parallel algorithms for the high-frequency MLFMA [6]–[11]. State-of-the-art implementations rely on a hierarchical distribution of radiation patterns in which radiation patterns, containing $\mathcal{O}(L^2)$ sampling points, are distributed among $P=\mathcal{O}(L^2)$ parallel processes [10]–[15]. This way, each process holds only $\mathcal{O}(1)$ sampling points in local memory. Consequently, to compute the translations in the MLFMA, each process requires only a corresponding subset of the translation operator.

We propose a novel algorithm based on the parallelization of the IM. The algorithm does require inter-process communication, however, the key result is that the translation operator is computed in $\mathcal{O}(\log L)$ time using $P=\mathcal{O}(L^2)$ parallel processes. For a realistic L and P, the proposed algorithm is roughly 10 times faster than a naive, baseline parallel algorithm.

Recently, a method to compute Legendre polynomials in a complexity of $\mathcal{O}(1)$, regardless of argument or degree, has been developed [16], in contrast to routines that are based on the well-known Legendre recursion formulas. As we will explain in this paper, this is essential for our proposed algorithm to obtain a good scaling behavior.

This paper is organized as follows: first, in Section II the notation is established and assumptions are stated. Section III describes the actual parallel algorithm and the computational complexity is derived. This theoretical work is validated by benchmarking an implementation of the algorithm in Section IV. Finally, in Section V, our conclusions are presented.

II. Problem Description and Preliminaries

A. Problem Description

For the high-frequency MLFMA, the required value for L to obtain a desired accuracy ϵ is given by [3]

$$L \approx \sqrt{3}ka + 1.8 \log_{10}^{2/3}(1/\epsilon)(\sqrt{3}ka)^{1/3} \tag{2}$$

where a denotes the box edge length. L roughly doubles at every next level up in the MLFMA-tree. As radiation patterns and translation operators are sampled in $\mathcal{O}(L^2)$ angular points, their sampling rate increases by a factor of approximately four at each higher level. Table I lists the runtime for the sequential computation of a single translation operator using both the DM and IM for different MLFMA-levels. The $\mathcal{O}(L^3)$ and $\mathcal{O}(L^2)$ time complexities for the DM and IM respectively are clearly observed.

In the hierarchical parallel MLFMA, the sampling points of the radiation patterns and translation operators are partitioned in an increasing number of $1, 4, 16, \ldots, 4^n$ parallel processes for every higher level in the MLFMA-tree [12], [13]. Formally stated, the $\mathcal{O}(L^2)$ angular sampling points are partitioned among $P = \mathcal{O}(L^2)$ parallel processes, such that each process contains $\mathcal{O}(1)$ sampling points in local memory. As both the problem size and the number of processes are proportionally increased with each MLFMA-level, the parallelization of the computation of the translation operator should be treated as a *weak scaling* parallelization problem. Therefore the main focus of the parallelization should be the weak scaling behavior of the algorithm, rather than its strong scaling behavior, i.e., the speedup for the computation of a fixed-size translation operator as a function of the number of processes. Throughout the whole paper, the term *scaling* refers to weak scaling.

A baseline, communication-free parallel algorithm for the distributed computation of the translation operator is easily obtained by trivially parallelizing the DM: each of the processes computes its own, local partition of the translation operator sampling points directly, using (1). This algorithm is referred to as the “parallel direct method” (PDM). It is “embarrassingly parallel” and hence it exhibits a very good parallel efficiency: the parallel speedup compared to the DM is almost equal to the number of parallel processes P (see Table I). However, the PDM has a time complexity of $\mathcal{O}(L)$, which is suboptimal. Even though the computation time of a single translation operator using the PDM is relatively modest, several thousands of translation operators need to be evaluated during the setup stage, making their calculation a considerable computational burden that takes hours for large-scale simulations.

We propose an algorithm that is based on the parallelization of the IM. Even though the algorithm is straightforward in concept and relatively easy to implement, the derivation of the computational complexity is intricate. Prior to describing the actual algorithm, some concepts and notations used through the remainder of this paper are introduced.

B. Preliminaries

There are two main assumptions in this work. First, we assume that the radiation patterns and translation operators are sampled in a uniform way along the two angular dimensions θ and ϕ [17]. Another popular sampling scheme is to sample uniformly in ϕ, while the θ-dimension is sampled according to a Gauss-Legendre quadrature rule [18]. Both the uniform and the Gauss-Legendre sampling scheme have the same minimum sampling rate and therefore they are approximately equally efficient to perform the integration on the Ewald sphere. The main motivation for a uniform sampling in the θ-direction is that the interpolations at the lowest levels in the tree of the parallel MLFMA can be performed using FFTs [10], [17]. These interpolations are fast and accurate up to machine precision. The choice for a Gauss-Legendre sampling in θ would only have a minor influence on the analysis and concepts presented in this paper. The mathematical details and derivations in the Appendix would be more complicated, but the analysis of the parallel algorithm would be fundamentally the same. Therefore, the proposed method is still applicable and useful for a Gauss-Legendre sampling scheme.

Second, we assume that the sampling points are partitioned among the parallel processes in both the θ- and ϕ-dimension (see Fig. 1 left), which is called “blockwise partitioning”. The main advantage of this way of partitioning is that for each process its rectangular, blockwise patch on the sphere contains $\mathcal{O}(1)$ sampling points [10], [11], [14], [15]. This leads to a parallel MLFMA for which the memory requirements, communication volume and computation time per process are bounded by $\mathcal{O}(\log N)$ [10].

Fig. 1 (right) depicts a geometrical representation of the translation operator. From (1), it follows that the translation operator is axisymmetric with respect to the translation direc-

TABLE I
Runtime to compute a translation operator for an increasing L ($\epsilon = 10^{-6}$) for the direct method (DM), interpolation method (IM) and parallel direct method (PDM).

ka	L	DM (s)	IM (s)	PDM (s)	P
80	170	0.97	0.57	0.26	4
160	316	6.07	2.07	0.41	16
320	604	41.91	7.37	0.73	64
640	1171	302.5	28.6	1.32	256
1280	2295	2268	106.4	2.57	1024
2560	4532	17422	436.6	5.04	4096

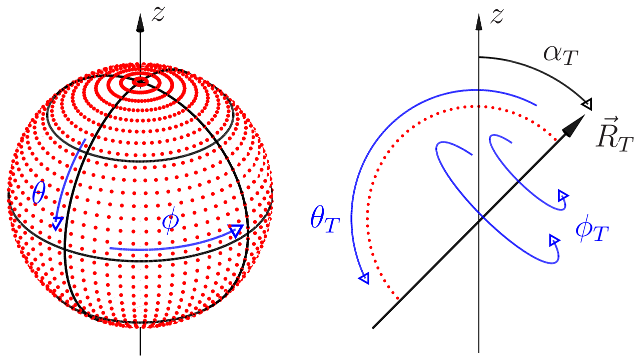

Fig. 1. Left: uniform sampling along the θ- and ϕ-direction. The dots correspond to the sampling points, while the solid lines denote the boundaries of the blockwise partitions assigned to different parallel processes (16 in this example). Right: geometrical representation of the translation operator along the $\vec{R}_T$-direction. The translation operator is axisymmetric with respect to $\vec{R}_T$, hence it depends only on θ_T and not on ϕ_T.

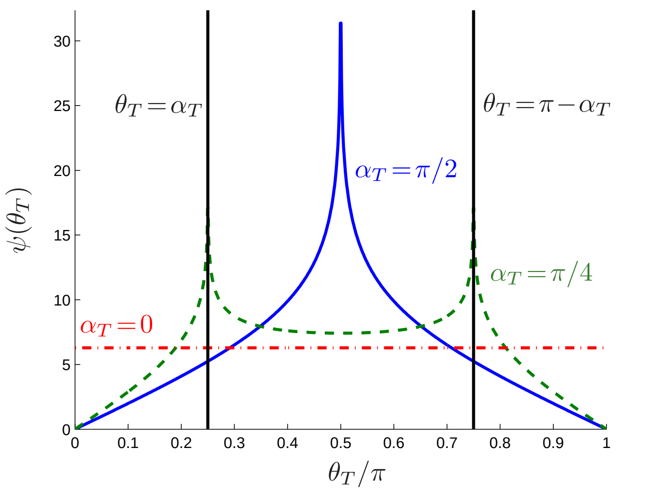

Fig. 2. Density function $\psi(\theta_T)$ for a number of angles: $\alpha_T = 0$ (red dash-dotted line), $\alpha_T = \pi/4$ (green dashed line) and $\alpha_T = \pi/2$ (blue solid line). The black vertical lines denote the logarithmic singularities at $\theta_T = \alpha_T$ and $\theta_T = \pi - \alpha_T$ for the case $\alpha_T = \pi/4$.

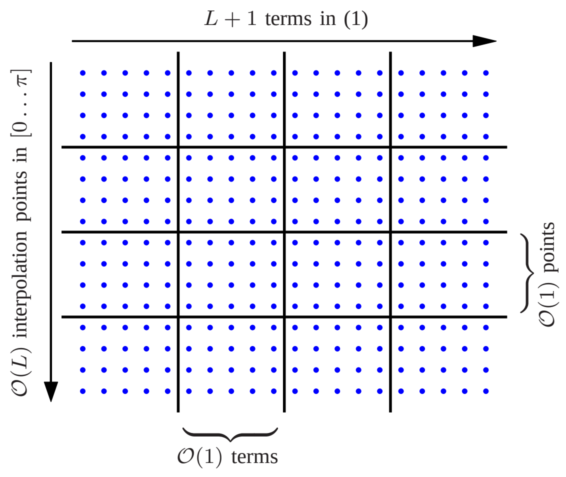

Fig. 3. Parallelization concept of the first step of the PIM algorithm: the $\mathcal{O}(L)$ interpolation points are distributed among $\sqrt{P} = \mathcal{O}(L)$ groups of parallel processes (horizontal solid lines). The computation of each of these interpolation points is further parallelized by splitting the $L+1$ terms between the $\sqrt{P}$ processes within each group (vertical solid lines). Each process hence computes $\mathcal{O}(1)$ terms for $\mathcal{O}(1)$ interpolation points. These terms are summed over all processes in a group using an all-reduce operation.

tion $\vec{R}_T$ and hence does not depend on ϕ_T. Because uniform sampling leads to an accumulation of sampling points at the poles of the sphere, the number of sampling points of the translation operator that needs to be evaluated is not uniform as a function of θ_T. Clearly, this distribution, referred to as the density function $\psi(\theta_T)$, depends only on the angle α_T between the z-axis and the translation direction $\vec{R}_T$ (see Fig. 1). In the Appendix, for the limit $L \to +\infty$, a closed-form expression for the density function is derived and the result is

$$\psi(\theta_T) = 4\frac{\sin(\theta_T)}{\sqrt{\beta}} K\left(2\sqrt{\frac{\sin(\theta_T)\sin(\alpha_T)}{\beta}}\right) \tag{3}$$

with

$$\beta = (1 + \sin(\alpha_T)\sin(\theta_T))^2 - (\cos(\alpha_T)\cos(\theta_T))^2 \tag{4}$$

and $K(k)$ the complete elliptic integral of the first kind. The Appendix also gives a convenient and easy way to numerically evaluate this special function.

The density function $\psi(\theta_T)$ is proportional to the number of sampling points of a translation operator that need to be evaluated in a particular θ_T-point. Expression (3) can be seen as the continuous approximation of the histogram that would represent this information for a finite L [19].

In Fig. 2, $\psi(\theta_T)$ is plotted for a number of different angles α_T. In the Appendix it is proven that, for all generic angles ($\alpha_T \neq 0$, $\alpha_T \neq \frac{\pi}{2}$ and $\alpha_T \neq \pi$), $\psi(\theta_T)$ has two logarithmic singularities at $\theta_T = \alpha_T$ and $\theta_T = \pi - \alpha_T$. For $\alpha_T = 0$ or $\alpha_T = \pi$, the translation operator direction $\vec{R}_T$ is parallel to the z-axis and, consequently, $\psi(\theta_T)$ is the uniform distribution. For $\alpha_T = \frac{\pi}{2}$, the translation direction $\vec{R}_T$ is perpendicular to the z-axis, resulting a single logarithmic singularity of $\psi(\theta_T)$ at $\theta_T = \alpha_T = \pi - \alpha_T = \frac{\pi}{2}$.

III. Parallel algorithm

This section discusses the different steps in the distributed-memory parallelization of the computation of the translation operator and their complexities. The proposed parallel method is essentially a parallel version of the two-step IM and is further referred to as the "parallel interpolation method" (PIM).

A. *Parallelization concept*

In short one can summarize the workflow of the sequential IM as presented in [4] as follows:

- Evaluate $T(\theta_T)$ in $\mathcal{O}(L)$ equidistant points in the interval $\theta_T \in [0\ldots\pi]$ using (1). We further refer to these points as interpolation points.
- Compute the translation operator in the required $\mathcal{O}(L^2)$ sampling points by local interpolation using the points from the previous step.

The number of source interpolation points needed to compute a single sampling point of the translation operator using local interpolation is $\mathcal{O}(1)$, i.e. independent of L [20]. Consequently, the time complexity for both steps is $\mathcal{O}(L^2)$.

The first step of the IM can be parallelized in a straightforward way. First, the P processes are subdivided in $\sqrt{P}$ groups, each consisting of $\sqrt{P}$ processes. As we are using $P = \mathcal{O}(L^2)$ parallel processes, this corresponds to $\mathcal{O}(L)$ groups, where each group contains $\mathcal{O}(L)$ processes. The $\mathcal{O}(L)$ interpolation points in the interval $[0\ldots\pi]$ are uniformly partitioned among these $\mathcal{O}(L)$ groups such that each group is responsible for the computation of $\mathcal{O}(1)$ interpolation points. Explicitly, $\theta_{T,p}$, the left boundary point of process group p, is determined by $\theta_{T,p} = \frac{p}{\sqrt{P}}\pi$ (with $p = 0\ldots\sqrt{P}-1$). For this step no communication is required *between* groups. The computations *within* a group can be further parallelized: each process evaluates and sums only a subset of the $L+1$ terms in (1) for each interpolation point in the group. This partitioning scheme is depicted in Fig. 3. The computations take $\mathcal{O}(1)$ time and there are no overlapping computations between processes. It is important to remark that we assume that spherical Hankel functions and Legendre polynomials can be effectively evaluated in $\mathcal{O}(1)$ time, regardless of the order l, ranging from 0 to L. For the spherical Hankel functions, the Amos library [21] can be used. For the Legendre polynomials, such a method was recently developed in [16].

To complete the first step, the partial results need to be summed over all processes within a group. This summation, which of course does require communication, is performed in such a way that the resulting sum for each interpolation point in the group is present in each process of the group. In parallel computing, this operation is referred to as an all-reduce operation, which can be performed in $\mathcal{O}(\log L)$ time [22]. As this is the dominating complexity, the first step of the PIM also requires $\mathcal{O}(\log L)$ time. At the end of the first step, each process contains the evaluated interpolation points that correspond to the group to which the process belongs. In other words, the calculated interpolation points are redundantly stored in each process of a group, however, the computations themselves are not duplicated between processes.

Conceptually, the parallelization of the second step of the IM is trivial: each process computes the translation operator in the required $\mathcal{O}(1)$ sampling points of its local blockwise partition by local interpolation using the points generated in step 1. As each of the $\mathcal{O}(1)$ sampling points of a process requires $\mathcal{O}(1)$ interpolation points, it follows that each process needs only $\mathcal{O}(1)$ interpolation points in total to perform this second step. No dependencies between computations exist and hence, no communication between processes is required.

However, prior to this second step, a mismatch exists between the subset of interpolation points that is required by a certain process to perform its computations in step 2 and the interpolation points that are actually present in local memory of that process at the end of step 1. Therefore a communication phase has to take place in between both steps in which the interpolation points are redistributed among the processes. As each process requires only $\mathcal{O}(1)$ interpolation points in step 2, it follows that the total volume of data received by any process during this reshuffling phase is also bounded by $\mathcal{O}(1)$. In the next section, we show that the total volume of data to be sent by a process is bounded by $\mathcal{O}(\log L)$.

B. Volume of data to be sent during the reshuffling phase

In step 1, the $\mathcal{O}(L)$ interpolation points in the interval $[0 \ldots \pi]$ are uniformly partitioned among the $\sqrt{P} = \mathcal{O}(L)$ groups. Fig. 4 illustrates the uniform partitioning for several density functions corresponding to different values of α_T.

Recall that the density function $\psi(\theta_T)$ is proportional to the number of sampling points of the translation operator that depend on the value of θ_T. As the density function is non-uniform in general, certain interpolation points are required by more sampling points (and hence more processes) than others, giving rise to non-uniform communication patterns during the reshuffling of interpolation points in between steps 1 and 2. To determine the communication complexity, we consider the worst-case scenario which occurs at the singularities of $\psi(\theta_T)$, i.e. at $\theta_T = \alpha_T$ and $\theta_T = \pi - \alpha_T$.

Consider the process group that contains the interpolation points around such a singularity, namely the interval $\theta_T =$

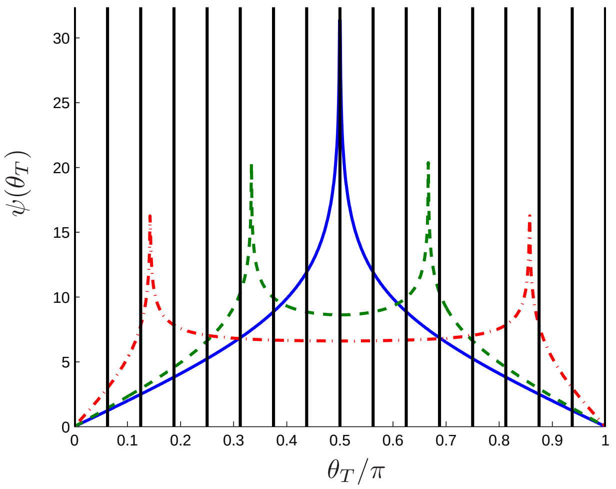

Fig. 4. Uniform partitioning of the interpolation points as a function of θ_T among $\sqrt{P}$ process groups (16 in this example). The vertical lines denote the partition boundaries. Additionally, the density function $\psi(\theta_T)$ is shown for $\alpha_T = \pi/2$ (blue solid line), $\alpha_T = \pi/3$ (green dashed line) and $\alpha_T = \pi/7$ (red dot-dashed line).

$[\alpha_T - \Delta\theta_T \ldots \alpha_T + \Delta\theta_T]$, with $\Delta\theta_T = \frac{\pi}{2\sqrt{P}}$. The total number of translation operator sampling points that correspond to these θ_T-values is proportional to

$$\begin{aligned}
&\sim L^2 \int_{\alpha_T-\Delta\theta_T}^{\alpha_T+\Delta\theta_T} \psi(\theta_T) d\theta_T && \text{(5a)}\\
&\simeq L^2 \int_{\alpha_T-\Delta\theta_T}^{\alpha_T+\Delta\theta_T} \left(C_1 \ln|\theta_T - \alpha_T| + C_2\right) d\theta_T && \text{(5b)}\\
&= 2L^2 \Delta\theta_T \left(-2\left(\ln(\Delta\theta_T) - 1\right) + C_2\right) && \text{(5c)}\\
&= \mathcal{O}(L \log L) && \text{(5d)}
\end{aligned}$$

where we used an approximation of $\psi(\theta_T)$ around $\theta_T = \alpha_T$ (derived in the Appendix) in (5b), $C_1 = -2$ in (5c) and the fact that $\Delta\theta_T = \mathcal{O}(\frac{1}{L})$ in (5d). From (5d) one sees that, in the worst-case scenario, there are $\mathcal{O}(L \log L)$ sampling points that depend on the θ_T-interval of a single process group.As there are $\sqrt{P} = \mathcal{O}(L)$ processes in that group that can deliver this data, no process has to send more than $\mathcal{O}(\log L)$ data in between steps 1 and 2 when these communications are equally divided among the $\sqrt{P}$ processes.

We conclude that the volume of data to be sent by any process in between the two steps is bounded by $\mathcal{O}(\log L)$.

C. Summary of the parallel algorithm

The steps of the parallel algorithm to calculate a translation operator can be summarized as follows:

- Step 1a: Assign interpolation points to each group of processes. Each process within each group computes only a subset of the terms of (1) for each of the interpolation points in the group. Cost: $\mathcal{O}(1)$.
- Step 1b: Perform the parallel summation (all-reduce operation) over all processes within each group. Cost: $\mathcal{O}(\log L)$.
- Reshuffling phase: Interpolation points are redistributed among processes. Cost: $\mathcal{O}(1)$ receive volume for each

process, $\mathcal{O}(\log L)$ send volume for the processes near the singularities of $\psi(\theta_T)$.
- Step 2: Compute the translation operator in its sampling points using local interpolation. Cost: $\mathcal{O}(1)$.

Assuming that all computations and communications by the different processes can be performed concurrently, the global complexity of the parallel algorithm is $\mathcal{O}(\log L)$.

IV. Numerical results

In this section, the implementation and the scaling behavior of the proposed PIM is numerically validated. The numerical data has been obtained using a cluster consisting of 256 machines each containing two 8-core Intel Xeon E5-2670 processors (4096 CPU-cores in total). The machines were connected using an FDR Infiniband network. To produce the results for $P = 16\,384$, each CPU-core has been oversubscribed by 4 processes. The calculations were performed in double-precision.

A. Validation of the implementation

To validate the implementation of the parallel computation of the proposed PIM algorithm, we consider the plane wave decomposition of the Green's function [2], [3]

$$\frac{1}{4\pi r}e^{-jkr} \simeq \int_0^{2\pi}\int_0^{\pi} T(\vec{R}_T,\theta,\phi)e^{-j\vec{k}\cdot\vec{R}_A}\sin(\theta)d\theta d\phi \quad (6)$$

with $\vec{k} = k(\cos\phi\sin\theta\,\vec{1}_x + \sin\phi\sin\theta\,\vec{1}_y + \cos\theta\,\vec{1}_z)$. The factor $e^{-j\vec{k}\cdot\vec{R}_A}$ is the aggregation, with $||\vec{R}_T + \vec{R}_A|| = r$. We chose $R_T = 3a$ and $\vec{R}_A = a(\vec{1}_x + \vec{1}_y + \vec{1}_z)$. For the target precision ϵ in (2) we consider three values: 10^{-3}, 10^{-6} and 10^{-9}. The translation operators are calculated for the same values of ka and P as in Table I, extended with $ka = 5120$ and $P = 16\,384$.
The total number of interpolation points in the interval $[0\ldots\pi]$ is set to $4L + 8$. The θ- and ϕ-dimensions are sampled in $L+1$ and $2L+4$ points respectively, and as a result the translation operators contain a total number of $2L^2 + 6L + 4$ sampling points. These sampling rates are realistic, as they correspond to the actual sampling rates that are used in our MLFMA simulations. The local interpolation method is based on the product of the Dirichlet kernel and a Gaussian function [3, p. 65]. The number of neighboring interpolation points is chosen sufficiently high, so that the local interpolation is accurate up to machine precision. This way, the error of the addition theorem is only determined by the value of L.

As discussed, the communication of the interpolation points in the PIM algorithm strongly depends on α_T and hence on the translation direction $\vec{R}_T$. Therefore we consider the following 26 translation directions

$$\vec{R}_T = R_T \cdot \frac{x\,\vec{1}_x + y\,\vec{1}_y + z\,\vec{1}_z}{\sqrt{x^2+y^2+z^2}} \quad (7)$$

where x, y and z take all combinations of the values -1, 0 and 1, except the case $x = y = z = 0$.

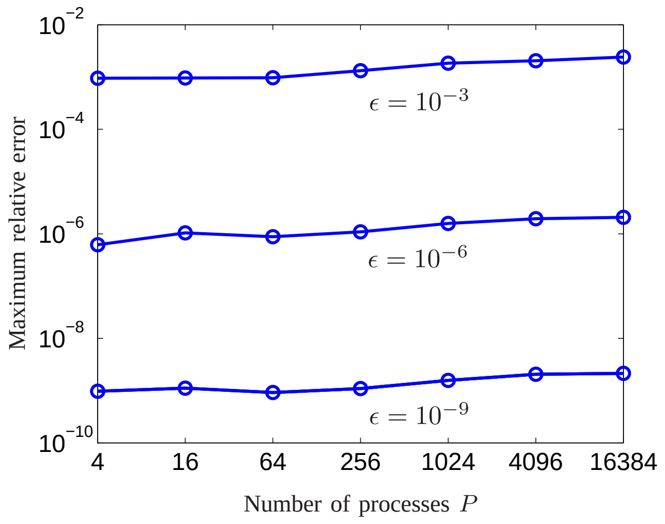

Fig. 5. Maximum relative error of the addition theorem as a function of partitions P.

TABLE II
RUNTIME TO COMPUTE A TRANSLATION OPERATOR FOR AN INCREASING L ($\epsilon = 10^{-6}$) FOR THE PROPOSED PIM ALGORITHM.

ka	L	PIM (s)	P
80	170	0.15	4
160	316	0.15	16
320	604	0.16	64
640	1171	0.18	256
1280	2295	0.27	1024
2560	4532	0.62	4096

Fig. 5 shows the maximum relative error of the addition theorem as a function of the number of partitions P. As one can see, the obtained precision of the worst-case translation direction corresponds well to the target precision. The translation operators produced by the PIM are identical to the ones obtained through the sequential IM.

B. Runtime benchmark

Table II shows the average runtime of the translation operators that were computed in the previous section, up to $P = 4096$. The values for L and P are the same as in Table I, and therefore the runtimes can be compared.

First, by comparing the values in Tables I and II, one observes that for high values of L the proposed PIM algorithm is roughly 10 times faster than the baseline PDM. This is a manifestation of the fact that the PIM has a lower time complexity, namely $\mathcal{O}(\log L)$, with respect to the $\mathcal{O}(L)$ complexity of the PDM. When considering problems with billions of unknowns, such as the one presented in [11], the runtime for the computation of all translation operators during the setup stage is reduced from approximately one hour (PDM) to only a few minutes (PIM). With an overall runtime (setup time + solution time) for this simulation of about 45 hours, it is clear that even though this reduction is important, the relative reduction in overall runtime is modest. However, because the proposed PIM reduces the computational complexity, we expect that this gain in performance will become relatively more important when even larger simulations are considered.

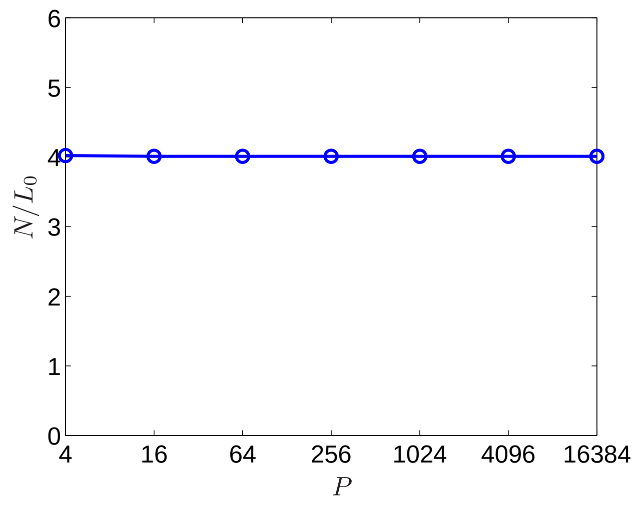

Fig. 6. Normalized number of interpolation points per process group for increasing P and L (with $L+1 \sim \sqrt{P}$).

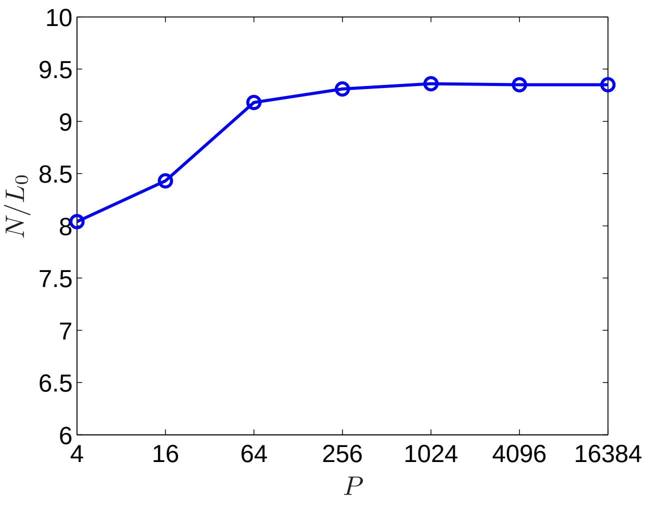

Fig. 7. Maximum normalized number of interpolation points required by a process in order to perform the local interpolation of its local translation sampling points.

Second, one can see that the runtime of the PIM increases faster than $\mathcal{O}(\log L)$ for a higher number of parallel processes P. This is caused by a limitation in the interconnection network of the cluster that was used. Specifically, the network supports only a limited number of concurrent communications between processes, which can cause a serialization of the communication and result in a slowdown of a factor two during the communication stage. Hence, we obtain runtimes for the PIM that are higher than expected, when using 1024 and 4096 processes.

C. Validation of the theoretical complexities

In this section we want to numerically validate the theoretically derived complexities of the PIM. The same translation directions of (7) are used and the values corresponding to the worst-case are selected, just as in Section IV-A, in which the implementation has been validated.

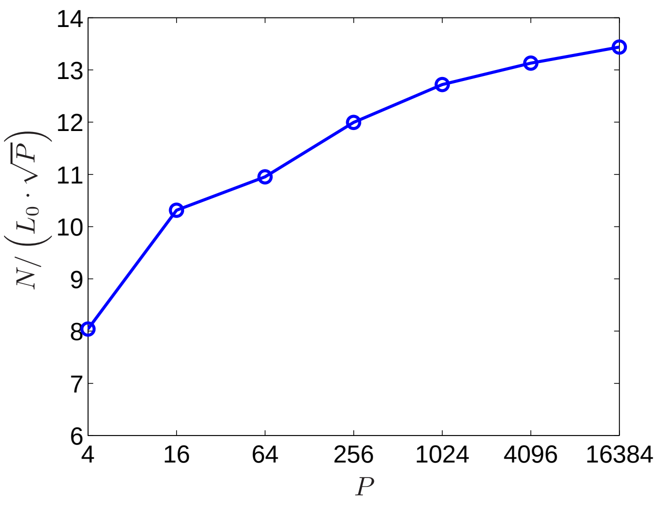

Fig. 8. Total number of interpolation points a process group has to send, normalized by L_0 and $\sqrt{P}$.

This time, the number of multipoles $L+1$ is set to $L_0 \cdot \sqrt{P}$, with $L_0 = 100$, instead of using (2). This way, a purely linear relationship between the number of multipoles and $\sqrt{P}$ is obtained, which corresponds to the high-frequency limit ($ka \gg 1$). As L_0 is purely arbitrary, the resulting number of interpolation points can be normalized with respect to L_0. By enforcing a purely linear dependency between $L+1$ and $\sqrt{P}$, the asymptotic behavior of the proposed PIM becomes apparent for lower values of L. Note that exactly the same conclusions will be obtained if (2) is used to calculate L.

Fig. 6 shows the number of interpolation points per process group as a function of P, normalized by L_0. As expected for uniform partitioning, this value is the same for each group and constant for an increasing number of P and L.

The maximum normalized number of interpolation points to be received by a process, in order to compute its local translation operator sampling points, is shown in Fig. 7. For increasing P and L it is bounded by $\mathcal{O}(1)$, as a result of the blockwise partitioning of the translation sampling points.

Fig. 8 displays the total number of interpolation points a process group has to send, divided by L_0 and $\sqrt{P}$, i.e. the number of processes a group contains. In case of uniform partitioning this is proportional to $\log P$ or, equivalently, $\log \sqrt{P} = \log L$, which corresponds exactly to the behavior predicted by the theory.

The results of this section show that the numerically obtained data corresponds very well to the theoretically predicted scaling behavior of the PIM.

V. Conclusion

In this paper the distributed-memory parallelization of the calculation of the translation operator in the MLFMA by means of the interpolation method was studied. To calculate a translation operator with $L+1$ multipoles using $P = \mathcal{O}(L^2)$ processes, our proposed algorithm requires only $\mathcal{O}(\log L)$ time, which is a clear improvement over the $\mathcal{O}(L)$ complexity of the baseline parallel algorithm. The average time to compute a translation operator using the parallel interpolation method is measured using 4096 CPU-cores and compared to a parallel implementation of the baseline method. As a result, a large speedup factor for realistic electromagnetic problems is achieved, which reduces the time of the setup stage significantly. Furthermore, the theoretical results for the parallel interpolation method were numerically verified using up to $16\,384$ processes and its scaling behavior corresponded very well to the theoretical analysis.

Acknowledgment

The computational resources (Stevin Supercomputer Infrastructure) and services used in this work were provided by Ghent University, the Hercules Foundation and the Flemish Government – department EWI.
The work of B. Michiels was supported by a doctoral grant from the Special Research Fund (BOF) at Ghent University. The work of I. Bogaert was supported by a post-doctoral grant from Research Foundation-Flanders (FWO-Vlaanderen).

Appendix

In this appendix an expression for the density function of the translation sampling points as a function of θ_T is derived.

A. Density function

From Fig. 1 one sees that there is an accumulation of sampling points at the poles of the sphere, due to the uniform sampling in θ and ϕ. As a result, one obtains a non-uniform distribution of the sampling points as a function of θ_T that depends on $\vec{R}_T$, the direction of the translation.

Consider the asymptotic case with $L \rightarrow +\infty$ and choose the (θ,ϕ)-coordinate system so that $\vec{R}_T = R_T(0, \sin(\alpha_T), \cos(\alpha_T))$, with $\alpha_T \in [0, \pi]$. As the Jacobian of a spherical coordinate system on the unit sphere is equal to $\sin(\theta)$ and as the sampling points are uniformly sampled in (θ, ϕ), the density of the sampling points $w(\theta, \phi)$ is

$$w(\theta,\phi) = \frac{1}{\sin(\theta)} \tag{8a}$$

$$= \frac{1}{\sqrt{1-\cos^2(\theta)}} \tag{8b}$$

Now consider the coordinate system of the translation direction, where the z-axis is parallel to $\vec{R}_T$. Using a rotation over the angle α_T about the x-axis, $\cos(\theta)$ can be expressed in this coordinate system as

$$\cos(\theta) = \sin(\alpha_T)\sin(\theta_T)\sin(\phi_T) + \cos(\alpha_T)\cos(\theta_T) \tag{9}$$

The density function of the translation sampling points as a function of θ_T is equal to

$$\psi(\theta_T) = \sin(\theta_T)\int_0^{2\pi} w(\theta,\phi) d\phi_T \tag{10}$$

as $\sin(\theta_T)$ is the radius of a circle of latitude in the coordinate system of $\vec{R}_T$. In the special case when $\alpha_T = 0$ or $\alpha_T = \pi$, $\psi(\theta_T)$ degenerates to

$$\psi(\theta_T) = \sin(\theta_T)\int_0^{2\pi} w(\theta,\phi) d\phi_T \tag{11a}$$

$$= \sin(\theta_T)\int_0^{2\pi} \frac{1}{\sin(\theta_T)} d\phi_T \tag{11b}$$

$$= 2\pi \tag{11c}$$

B. Elliptic integral

To find an expression for the density function $\psi(\theta_T)$ one has to calculate the integral

$$I = \int_0^{2\pi} \frac{1}{\sqrt{1-(a\sin(\phi)+b)^2}} d\phi \tag{12}$$

with $a = \sin(\alpha_T)\sin(\theta_T)$ and $b = \cos(\alpha_T)\cos(\theta_T)$.
As $\sin(\phi)$ ranges from -1 to $+1$ in the interval $\phi = -\frac{\pi}{2} \ldots \frac{\pi}{2}$, one can write

$$I = 2\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{1}{\sqrt{1-(a\sin(\phi)+b)^2}} d\phi \tag{13}$$

The key to simplify the integrand is the substitution

$$x = \sqrt{\frac{1-b+a}{1-b-a} \cdot \frac{1-\sin(\phi)}{1+\sin(\phi)}} \tag{14}$$

The variable x is a positive real number, as

$$b \pm a = \cos(\alpha_T \mp \theta_T) \tag{15}$$

is smaller than 1 when $\alpha_T \neq \theta_T$. For $\phi = -\frac{\pi}{2} \ldots \frac{\pi}{2}$, x ranges from $+\infty$ to 0.
Finally, after numerous yet straightforward algebraic operations, one obtains

$$I = 4\int_0^{+\infty} \frac{1}{\sqrt{\gamma_+}} \cdot \frac{1}{\sqrt{(1+x^2)\left(1+\frac{\gamma_-}{\gamma_+}x^2\right)}} dx \tag{16a}$$

$$= \frac{4}{\sqrt{\gamma_+}} K\left(\sqrt{\frac{4a}{\gamma_+}}\right) \tag{16b}$$

with $\gamma_\pm = (1 \pm a)^2 - b^2$ and $K(k)$ the complete elliptic integral of the first kind [23]. Using (10) and the definitions of a and b, one finds the expression of (3).
For the special cases $\alpha_T = 0$ and $\alpha_T = \pi$ one sees that

$$a = 0 \tag{17a}$$

$$b = \pm\cos(\theta_T) \tag{17b}$$

$$I = \frac{2\pi}{\sin(\theta_T)} \tag{17c}$$

as $K(k = 0) = \frac{\pi}{2}$. This result corresponds to the result of (11).

C. Singularities

The complete elliptic integral of the first kind has a logarithmic singularity in $k = 1$ [23]:

$$k \to 1: \quad K(k) \simeq \ln\left(\frac{1}{\sqrt{1-k^2}}\right) + 2\ln(2) \tag{18}$$

From (3) one can derive that the density function $\psi(\theta_T)$ has a logarithmic singularity in $\theta_T = \alpha_T$ and $\theta_T = \pi - \alpha_T$, except for the degenerate cases $\alpha_T = 0$ and $\alpha_T = \pi$.

Assume that α_T is not very close to 0 or π. After a second order Taylor expansion of the argument of the elliptic integral around $\theta_T = \alpha_T$ and substituting $\theta_T = \alpha_T$ in the non-singular part of $\psi(\theta_T)$ one obtains

$$\psi(\theta_T) \simeq -2\ln\left|\frac{1}{2}\cot(\alpha_T)\left(\theta_T - \alpha_T\right)\right| + 4\ln(2) \tag{19a}$$

$$\simeq C_1 \ln|\theta_T - \alpha_T| + C_2 \tag{19b}$$

with

$$C_1 = -2 \tag{20a}$$

$$C_2 = 2\ln|8\tan(\alpha_T)| \tag{20b}$$

The approximation of (19) is valid as long as $|\theta_T - \alpha_T| \ll \alpha_T$. For the singularity in $\theta_T = \pi - \alpha_T$ one can derive a similar expression.

D. Numerical evaluation

The appearance of an elliptic integral in the expression of the density function $\psi(\theta_T)$ does not pose a problem because it can be easily and quickly computed using

$$K(k) = \frac{\pi}{2M(1-k, 1+k)} \tag{21}$$

with M the arithmetic-geometric mean [23].
For completeness we also mention that one should be very careful when evaluating $K(k)$ close to its singularity at $k = 1$ as the expression of (16b) would lead to numerical inaccuracies. To understand this we consider

$$1 - k = 1 - \sqrt{\frac{4a}{\gamma_+}} \tag{22a}$$

$$= \frac{1 - \frac{4a}{\gamma_+}}{1 + \sqrt{\frac{4a}{\gamma_+}}} \tag{22b}$$

$$= \frac{\gamma_-}{\gamma_+\left(1 + \sqrt{\frac{4a}{\gamma_+}}\right)} \tag{22c}$$

with

$$\gamma_- = (1 - a + b)(1 - a - b) \tag{23a}$$

$$= (1 + \cos(\alpha_T + \theta_T))(1 - \cos(\alpha_T - \theta_T)) \tag{23b}$$

When θ_T is close to α_T one can use a second order Taylor expansion

$$\cos(\alpha_T - \theta_T) \simeq 1 - \frac{1}{2}(\alpha_T - \theta_T)^2 \tag{24}$$

Suppose a machine precision δ. When θ_T is in the region of $\alpha_T \pm \sqrt{\delta}$, the evaluation of γ_- already reaches machine precision. If θ_T comes closer to α_T, the second factor of (23b) will be rounded to 0 or δ. This is clearly undesirable for the computation of the arithmetic-geometric mean $M(1-k, 1+k)$ and will lead to numerical inaccuracies in the calculation of the elliptic integral $K(k)$.
Therefore the numerator of (22c) has to be rewritten as

$$\gamma_- = 4\cos^2\left(\frac{\alpha_T + \theta_T}{2}\right)\sin^2\left(\frac{\alpha_T - \theta_T}{2}\right) \tag{25}$$

This expression does not suffer from a numerical breakdown and it allows to obtain accurate results when computing $K(k)$ numerically.

References

[1] L. Greengard and V. Rokhlin, "A Fast Algorithm for Particle Simulations", Journal of Computational Physics, vol. 73, no. 2, pp. 325–348, December 1987.

[2] R. Coifman, V. Rokhlin and S. Wandzura, "The Fast Multipole Method for the Wave Equation: A Pedestrian Prescription", IEEE Antennas and Propagation Magazine, vol. 35, no. 3, pp. 7–12, June 1993.

[3] W.C. Chew, J. Jin, E. Michielssen and J. Song, "Fast and Efficient Algorithms in Computational Electromagnetics", Artech House, 2001.

[4] J. Song and W.C. Chew, "Interpolation Of Translation Matrix In MLFMA", Microwave And Optical Technology Letters, vol. 30, no. 2, pp. 109–114, July 2001.

[5] Ö. Ergül and L. Gürel, "Optimal Interpolation of Translation Operator in Multilevel Fast Multipole Algorithm", IEEE Transactions on Antennas and Propagation, vol. 54, no. 12, pp. 3822–3826, December 2006.

[6] L. Gürel and Ö. Ergül, "Hierarchical Parallelization of the Multilevel Fast Multipole Algorithm (MLFMA)", Proceedings of the IEEE, vol. 101, no. 2, pp. 332–341, February 2013.

[7] Ö. Ergül and L. Gürel, "Accurate Solutions of Extremely Large Integral-Equation Problems in Computational Electromagnetics", Proceedings of the IEEE, vol. 101, no. 2, pp. 342–349, February 2013.

[8] J.M. Taboada, M.G. Araujo, F. Obelleiro, J.L. Rodriguez and L. Landesa, "MLFMA-FFT Parallel Algorithm for the Solution of Extremely Large Problems in Electromagnetics", Proceedings of the IEEE, vol. 101, no. 2, pp. 350–363, February 2013.

[9] X.M. Pan, W.C. Pi, M.L. Yang, Z. Peng and X.Q. Sheng, "Solving Problems with over One Billion Unknowns by the MLFMA", IEEE Transactions on Antennas and Propagation, vol. 60, no. 5, pp. 2571–2574, May 2012.

[10] B. Michiels, J. Fostier, I. Bogaert and D. De Zutter, "Weak Scalability Analysis of the Distributed-Memory Parallel MLFMA", IEEE Transactions on Antennas and Propagation, vol. 61, no. 11, pp. 5567–5574, Nov. 2013.

[11] B. Michiels, J. Fostier, I. Bogaert and D. De Zutter, "Full-wave simulation of electromagnetic scattering problem with more than three billion unknowns", IEEE Transactions on Antennas and Propgation, submitted for publication, 2014.

[12] Ö. Ergül and L. Gürel, "Hierarchical Parallelisation Strategy for Multilevel Fast Multipole Algorithm in Computational Electromagnetics", Electronics Letters, vol. 44, no. 1, pp. 3–4, January 2008.

[13] Ö. Ergül and L. Gürel, "A Hierarchical Partitioning Strategy for an Efficient Parallelization of the Multilevel Fast Multipole Algorithm", IEEE Transactions on Antennas and Propagation, vol. 57, no. 6, pp. 1740–1750, June 2009.

[14] J. Fostier and F. Olyslager, "Provably Scalable Parallel Multilevel Fast Multipole Algorithm", Electronics Letters vol. 44, no. 19, pp. 1111–1112, September 2008.

[15] B. Michiels, J. Fostier, I. Bogaert, P. Demeester and D. De Zutter, "Towards a scalable parallel MLFMA in three dimensions", Proceedings of the Computational Electromagnetics International Workshop (CEM '11), Izmir, pp. 132-135, Izmir, Turkey, August 2011.

[16] I. Bogaert, B. Michiels and J. Fostier, "$\mathcal{O}(1)$ Computation Of Legendre Polynomials And Gauss-Legendre Nodes And Weights For Parallel Computing", SIAM Journal on Scientific Computing, vol. 34, no. 3, pp. C83–C101, 2012.

[17] J. Sarvas, "Performing Interpolation and Anterpolation entirely by Fast Fourier Transform in the 3-D Multilevel Fast Multipole Algorithm", SIAM Journal on Numerical Analysis, 41(6):2180–2196, 2003.

[18] L. Gürel and Ö. Ergül, “Fast and accurate solutions of extremely large integral-equation problems discretised with tens of millions of unknowns”, Electronics Letters, vol. 43, no. 9, pp. 499–500, April 2007.
[19] B. Michiels, I. Bogaert, J. Fostier and D. De Zutter, “A Weak Scalability Study of the Parallel Computation of the Translation Operator in the MLFMA”, Proceedings of the International Conference on Electromagnetics in Advanced Applications (ICEAA 2013), pp. 369–399, Turin, Italy, September 2013.
[20] O.M. Bucci, C. Gennareli and C. Savarese, “Optimal Interpolation of Radiated Fields Over a Sphere”, IEEE Transactions on Antennas and Propagation, vol. 39, no. 11, pp. 1633–1643, November 1991.
[21] D.E. Amos, “A portable package for Bessel functions of a complex argument and nonnegative order”, ACM Transactions on Mathematical Software (TOMS), vol. 12, no. 3, pp. 265–273, September 1986.
[22] R. Thakur and R. Rabenseifner, “Optimization of collective communication operations in MPICH”, International Journal of High Performance Computing Applications, vol. 19, pp. 49–66, 2005.
[23] F. Olver, D. Lozier, R. Boisvert and C. Clark, “NIST Handbook Of Mathematical Functions”, Cambridge University Press, New York, 2010.

Bart Michiels was born on October 30, 1986. He received the M.S. degree in engineering physics from Ghent University, Ghent, Belgium, in 2009. His master’s thesis dealt with the simulation of large, broadband two-dimensional electromagnetic problems, using the Multilevel Fast Multipole Method (MLFMA). In 2013 he obtained the Ph.D. degree in engineering physics at the Department of Information Technology (INTEC) in Ghent. The main topic of his Ph.D. thesis was the parallelization of the MLFMA and its parallel scalability, in order to solve large-scale full-wave electromagnetic problems.

Ignace Bogaert was born in Ghent, Belgium, in 1981. He received the M.S. degree in engineering physics from Ghent University, Ghent, Belgium, in 2004. After graduating, he joined the Electromagnetics Group of the Department of Information Technology (INTEC) at Ghent University, where he received his Ph.D. in applied physics in 2008. His research is supported by a postdoctoral grant from the Research Foundation-Flanders (FWO-Vlaanderen). His research interests include optimization problems and the modeling of various physical systems, with the emphasis on robustness, efficiency and accuracy.

Jan Fostier was born in 1982. He received the M.S. degree in Physical Engineering in 2005 and the Ph.D. degree in applied physics in 2009, both from Ghent University, Ghent, Belgium. In 2005, he joined the electromagnetics research group at the Department of Information Technology (INTEC) and in 2011 he was appointed assistant professor at the Internet Based Communication Networks and Services (IBCN) research group at the same department. His current research interests are numerical techniques, fast algorithms and distributed computing for bio-informatics and electromagnetics problems.

Daniël De Zutter was born in 1953. He received his M.Sc. Degree in electrical engineering from the University of Gent in 1976. In 1981 he obtained a Ph.D. degree and in 1984 he completed a thesis leading to a degree equivalent to the French Aggrégation or the German Habilitation (both at the University of Gent). He is now a full professor of electromagnetics. His research focusses on all aspects of circuit and electromagnetic modelling of high-speed and high-frequency interconnections and packaging, on Electromagnetic Compatibility (EMC) and numerical solutions of Maxwells equations. As author or co-author he has contributed to more than 200 international journal papers (cited in the Web of Science) and 200 papers in conference proceedings. In 2000 he was elected to the grade of Fellow of the IEEE. He was an Associate Editor for the IEEE Microwave Theory and Techniques Transactions. Between 2004 and 2008 he served as the Dean of the Faculty of Engineering of Ghent University and is now the head of the Department of Information Technology.