Parallel, Distributed-Memory Computation of the Translation Operator in Three Dimensions

Bart Michiels¹, Ignace Bogaert¹, Jan Fostier¹, Daniël De Zutter¹
¹Department of Information Technology (INTEC), Ghent University, Belgium

Abstract—A new algorithm for the parallel, distributed-memory computation of the translation operator in the three-dimensional (3D) Multilevel Fast Muiltipole Algorithm (MLFMA) is presented. In the MLFMA, translation operators with \( L \) multipoles need to be evaluated in \( \mathcal{O}(L^2) \) directions. The key property of the proposed algorithm is that such translation operators are evaluated in only \( \mathcal{O}(\log L) \) time, using \( P = \mathcal{O}(L^2) \) parallel processes. This relationship between \( P \) and \( L \) occurs naturally in parallel MLFMA implementations that rely on a hierarchical distribution of radiation pattern sampling points and their associated radiation patterns. Numerical results show that the proposed algorithm outperforms a baseline parallel algorithm by a factor of ten.

Index Terms—translation operator, parallel computing, MLFMA.

I. INTRODUCTION

In computational electromagnetics, the Multilevel Fast Multipole Algorithm (MLFMA) is one of the most effective ways to reduce the computational costs and memory requirements of the method. In this paper, we present a new algorithm for the parallel, distributed-memory computation of the translation operator in the MLFMA in three dimensions.

In the MLFMA, translation operators with \( L \) multipoles need to be evaluated in \( \mathcal{O}(L^2) \) angular directions. The key property of the proposed algorithm is that such translation operators are evaluated in only \( \mathcal{O}(\log L) \) time, using \( P = \mathcal{O}(L^2) \) parallel processes. This relationship between \( P \) and \( L \) occurs naturally in parallel MLFMA implementations that rely on a hierarchical distribution of radiation pattern sampling points and their associated radiation patterns. Numerical results show that the proposed algorithm outperforms a baseline parallel algorithm by a factor of ten.

II. OUTLINE OF THE PARALLEL ALGORITHM

In order to parallelize the first step of the interpolation method, the \( P = \mathcal{O}(L^2) \) processes are dived in \( \sqrt{P} = \mathcal{O}(L) \) groups each group containing \( \sqrt{P} = \mathcal{O}(L) \) processes. Each group is responsible for the computation of a \( \mathcal{O}(1) \) operator of the interpolation points in the \([0 \ldots \pi]\) interval. This means that each group is assigned \( \mathcal{O}(1) \) interpolation points. The \( L + 1 \) terms from formula (1) are partitioned among the processes.
within a group and evaluated in parallel. For this, we rely on a recent method that computes the Legendre polynomial of arbitrary order in constant time [6]. These partial results are then summed over the processes such that each process within a group holds the final result (allreduce operation). The first step requires $O(1)$ compute time per process and $O(\log L)$ time for the allreduce operation.

The parallelization of the second step is trivial, as each process computes its local subset of the required translation operator directions using the interpolation points obtained in the previous step. This takes $O(1)$ time per process. However, these interpolation points might be computed by a different process group and hence not be locally available. Therefore, in between both steps, the interpolation points need to be shuffled between different processes. This communication phase is non-trivial and special care must be taken to ensure that communications are evenly spread among the different processes. It can be shown that the communication volume (sending and receiving) per process does not exceed $O(\log L)$ [5]. Assuming a parallel system where all processes can communicate in parallel (a non-blocking interconnection network), these communications can be completed in $O(\log L)$ time as well. Therefore, the complete PIM algorithm has a complexity of $O(\log L)$ parallel time.

### III. Numerical Results

All numerical data was 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. Fig. 1 depicts the runtime to compute a single translation operator for different levels of the MLFMA tree, and hence, different numbers of multipoles $L$. The values of $L$ were obtained by the excess bandwidth formula. Fig. 1 shows four methods: the sequential DM and IM and the parallel PDM and PIM methods. For the parallel methods, the number of parallel processes increases as $O(L^2)$, as encountered in parallel hierarchical MLFMA algorithms [7]. The computational complexities of $O(L^3)$, $O(L^2)$, $O(L)$ and $O(\log L)$ for the DM, IM, PDM and PIM respectively can be observed. For example, the proposed PIM algorithm outperforms the baseline PDM method by a factor of ten on average. As a consequence, for very large-scale parallel MLFMA simulations with hundreds of millions of unknowns, the PIM reduces the setup time of the MLFMA with more than one hour, compared to the PDM. Note that the PIM achieves exactly the same accuracy as its sequential IM counterpart.

### IV. Conclusion

We propose a parallel, distributed-memory algorithm for the computation of the translation operators in the three-dimensional MLFMA based on a parallelization of the interpolation method. The method computes a translation operator with $L$ multipoles in $O(L^2)$ angular directions using $P = O(L^2)$ parallel processes in only $O(\log L)$ time, a problem that arises naturally in hierarchical parallel MLFMA implementations. This improves upon a baseline parallel algorithm with a complexity of $O(L)$. The algorithm has been validated using 4096 parallel processes and removes a bottleneck in the setup stage of the MLFMA when dealing with an extremely large number of unknowns.

### 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).

### REFERENCES


