Solutions of Large Problems of Computational Electromagnetics at Bilkent University†

Levent Gürel[1,2], Özgür Ergül[1], and Tahir Malas[1]

[1]Department of Electrical and Electronics Engineering
[2]Computational Electromagnetics Research Center (BiLCEM)
Bilkent University, TR-06800, Bilkent, Ankara, Turkey
E-mail: lgurel@bilkent.edu.tr

Abstract
We present our efforts to solve large numerical problems, which are obtained from mathematical formulations of real-life electromagnetic problems. For the iterative solutions of dense matrix equations resulting from the formulations by the surface integral equations, we utilize advances in both solution algorithms and computer hardware. We employ parallel multilevel fast multipole algorithm (MLFMA) on a cluster of relatively inexpensive processors connected via special fast networks. Since the parallelization of MLFMA is not trivial, we consider different strategies to obtain the solutions with high efficiency. Various preconditioning schemes are also investigated to increase convergence rates, especially for the solutions of ill-conditioned matrix equations that are obtained from the electric-field integral equation.

1. Surface Integral Equations

There are three different surface integral equations to formulate the radiation and scattering problems [1]. These are the electric-field integral equation (EFIE), magnetic-field integral equation (MFIE), and combined-field integral equation (CFIE) defined as

$$\text{CFIE} = \alpha\text{EFIE} + (1-\alpha)\text{MFIE}, \tag{1}$$

where α is a constant between 0 and 1. CFIE is usually preferred over EFIE and MFIE for closed surfaces mainly because it is free of the internal-resonance problem [2] and it generates better-conditioned matrix equations that are crucial for contemporary iterative solvers [3], such as multilevel fast multipole algorithm (MLFMA) [4]. For open surfaces, MFIE and thus CFIE are not applicable, leaving EFIE as the only choice. Unfortunately, EFIE usually produces ill-conditioned matrix equations that are difficult to solve iteratively.

By the simultaneous discretization of integral equations and the geometry, we obtain $N \times N$ matrix equations, i.e,

$$\sum_{n=1}^{N} Z_{mn}^{E,M,C} a_n = v_m^{E,M,C}, \qquad m = 1, ..., N. \tag{2}$$

In the above, a_n represents the unknown coefficients of the basis functions. In addition, $Z_{mn}^{E,M,C}$ and $v_m^{E,M,C}$ represent the elements of the impedance matrix and the excitation vector,

†This work was supported by the Scientific and Technical Research Council of Turkey (TUBITAK) under Research Grant 105E172, by the Turkish Academy of Sciences in the framework of the Young Scientist Award Program (LG/TUBA-GEBIP/2002-1-12), and by contracts from ASELSAN and SSM

respectively. Matrix equations in (2) are solved iteratively, where the matrix-vector multiplications are accelerated by MLFMA. The next section summarizes the steps of MLFMA for the efficient solutions of dense matrix equations.

2. Structure of MLFMA

Using the physical setting of the method of moments, a matrix-vector multiplication can be regarded as a set of interactions between the basis and testing functions. In order to perform these interactions in a group-by-group manner, the whole geometry is placed into a large cube and it is recursively divided into smaller ones until the smallest cubes contain only a few basis or testing functions. This way, a tree structure is formed, which provides a hierarchical representation of the computational domain. Nonempty cubes in the MLFMA tree are referred to as clusters. The fundamental idea in MLFMA is to replace element-to-element interactions with cluster-to-cluster interactions in a multilevel scheme. This computational scheme relies on the factorization of the Green's function, which is valid only for basis and testing functions that are far from each other. In the lowest level, interactions between the near-field clusters are computed directly and stored in a sparse matrix. Interactions among the far-field clusters are computed approximately using MLFMA, which consists of three main steps [4].

1) Aggregation: The radiated fields of each cluster are aggregated at the centers of the clusters.
2) Translation: For each pair of far-field clusters, whose parents are near to each other, cluster-to-cluster interaction is computed via a translation.
3) Disaggregation: Matrix-vector multiplication is completed by disaggregating the incoming fields to the centers of the testing clusters and onto the testing functions.

MLFMA performs matrix-vector multiplications related to an $N \times N$ dense matrix equation with $\mathcal{O}(N \log N)$ processing time and $\mathcal{O}(N \log N)$ memory requirement. Therefore, sequential MLFMA employed in an iterative method provides the solution of very large electromagnetic scattering and radiation problems. However, it is also desirable to parallelize MLFMA to further increase the dimensions of the problems solvable on relatively inexpensive platforms, such as clusters of personal computers. On the other hand, parallelization of MLFMA is not trivial. Communications between the processors and duplications of the computations reduce the efficiency of the parallelization. In the following section, we provide the basic steps for the parallelization of MLFMA and outline our efforts to obtain a scalable algorithm.

3. Parallelization of MLFMA

An MLFMA implementation can be divided into two parts. In the first part, the required data for the matrix-vector multiplications are computed and stored in the memory. The data include the tree structure and clustering of MLFMA, near-field interactions, Fourier transforms for the basis and testing functions, and translation matrices for cluster-cluster interactions. In the second part, the problem is solved by an iterative algorithm employing MLFMA for matrix-vector multiplications that are performed as

$$\overline{\boldsymbol{Z}} \cdot \boldsymbol{x} = \overline{\boldsymbol{Z}}^{NF} \cdot \boldsymbol{x} + \overline{\boldsymbol{Z}}^{FF} \cdot \boldsymbol{x} \tag{3}$$

where the near-field interactions denoted by $\overline{\boldsymbol{Z}}^{NF}$ are calculated directly and stored in the memory. Since the number of near-field interactions varies for different rows of the matrix, we employ a load-balance algorithm to distribute the rows among the processors. This way, the interactions are shared equally among the processors and each processor calculates its own interactions independently during the setup. This is essential since the setup for the near-field interactions requires considerable time. On the other hand, for the far-field interactions denoted by $\overline{\boldsymbol{Z}}^{FF}$, the rows are distributed among the processors in a different way compared to the near-field partitioning. In addition, the iterative method is also parallelized according to the far-field partitioning. Consequently, we employ gather-scatter operation in each matrix-vector multiplication to match different partitioning of near-field and far-field interactions. After the input vector $\boldsymbol{x}$ is multiplied with the near-field matrix in each processor separately, the output vector with the near-field partitioning is collected and redistributed to the processors for the far-field calculations and the iterative method. The additional cost due to the gather-scatter operation is usually negligible compared to the processing time gained by the load-balancing of the near-field matrix.

In order to evaluate the matrix-vector multiplication related to $\overline{\boldsymbol{Z}}^{FF}$ in (2), aggregation, translation, and disaggregation steps are performed on the tree structure. In a simple parallelization of MLFMA, the clusters are distributed and assigned to the processors. In this case, a level, i.e., level of distribution (LoD), is chosen to share the clusters equally among the processors. If a cluster belongs to a processor, then all of its sub-clusters in the lower levels are also assigned to the same processor. This way, the aggregation and disaggregation steps from the bottom of the tree up to the LoD can be performed without any communication. On the other hand, for the upper levels above the LoD, a cluster and its parent cluster may belong to different processors. As a consequence, aggregation and disaggregation steps involve communications for high levels of the tree structure. To avoid these communications, we allow the clusters in the levels above the LoD to be duplicated in different processors, if required. This way, all the aggregation and disaggregation steps can be performed without any communication. When the radiated fields are translated into incoming fields, the contributions from different processors are combined automatically for the duplicated clusters.

In MLFMA, translations are required to convert the radiated fields calculated during the aggregation process into incoming fields to be used in the disaggregation process. In the simple parallelization of MLFMA, the translations are important since they easily become the bottleneck of the program. Some of the translations are related to the clusters that are assigned to the same processor and they can be completed without any communication. However, there are many other translations that require dense communications since they are related to the interactions of pairs of clusters, which belong to different processors. In general, each processor has radiation data to be sent to all other processors. We carefully handle the data traffic by using a fixed communication map between the processors. For p processors, all of the communications are completed in about p steps, each of which includes simultaneous exchanges of data between pairs of coupled processors. When the processors are coupled, data set of a cluster is transferred from the sender to the receiver if it is required. After a transfer, the receiver processor completes all the translations related to the transferred data. This way, it becomes possible to avoid transferring the same data more than once.

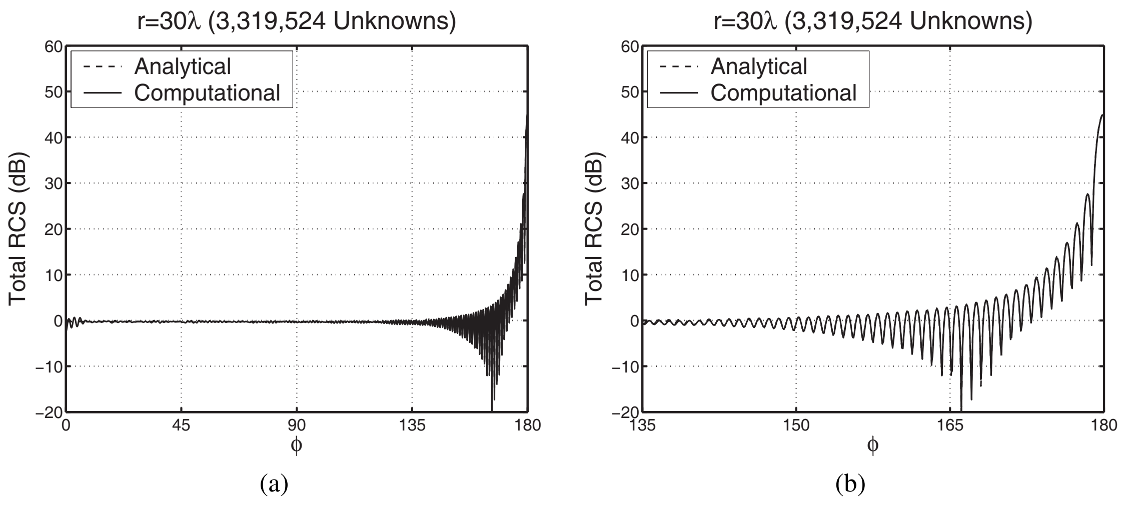
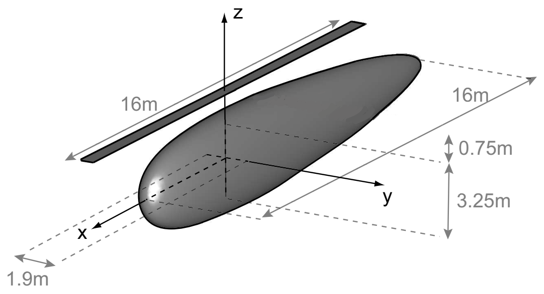


Fig. 1. Bistatic RCS in dB of a sphere of radius $a = 30\lambda$ calculated on the E-plane with respect to bistatic direction (a) from 0° (backscattering) to 180° (forward-scattering) and (b) from 135° to 180°. Analytical Mie-series results are also plotted to check the accuracy of the computations.

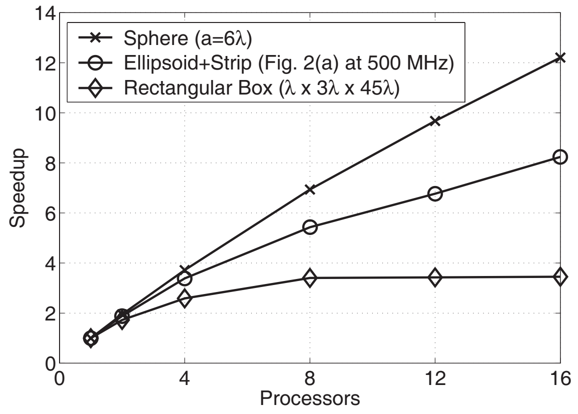
As an example to the solutions of large problems, Fig. 1 presents the results of a scattering problem related to a sphere of radius 30λ. The sphere is illuminated by a plane wave and Fig. 1(a) depicts the bistatic radar cross section (RCS) in decibels (dB) on the E-plane. For a clear comparison, the same data is plotted from 135^o to 180^o in Fig. 1(b). Both figures also include the analytical values obtained by a Mie series solution to show that the computational results are accurate. The MLFMA solution is performed on an 8-way SMP server with dual-core AMD Opteron processors. Using CFIE with block-diagonal preconditioner (BDP), residual error drops under 10^{-6} in only 29 iterations. The solution requires maximum 1750 MBytes and completed in about 8700 seconds.

Although a simple parallelization of MLFMA is relatively successful for the solution of smooth geometries, such as a sphere, it usually fails to provide an acceptable speedup for the solution of those having geometries that results in unbalanced tree structure. Most of the real-life problems is nonsymmetric and has a longer dimension in one direction so that their tree structures have only a few clusters in the upper levels. Then, the distribution of the tree structure among the processors becomes difficult and the efficiency of the parallelization drops significantly. As an example, Fig. 2(b) presents the speedup obtained by the simple parallelization of MLFMA for the solutions of three different scattering problems. Although the speedup is over 12 for 16 processors for the solution of a sphere problem with 132,003 unknowns, it is nearly 8 for the geometry described in Fig. 2(a) modelled by 117,366 unknowns and solved at 500 MHz. The speedup further decreases in the solution of a long rectangular box with dimensions of $\lambda \times 3\lambda \times 45\lambda$ and it is only about 4 for 16 processors.

It is obvious that a simple parallelization of MLFMA fails to provide an efficient solution for the problems with arbitrary geometries. For these problems, there are a few large clusters in the upper levels so that they cannot be shared among the processors with a good load balance. As a consequence, the aggregation and disaggregation steps cannot be performed efficiently. Therefore, in order to improve the load-balancing, we have developed a parallel



(a)



(b)

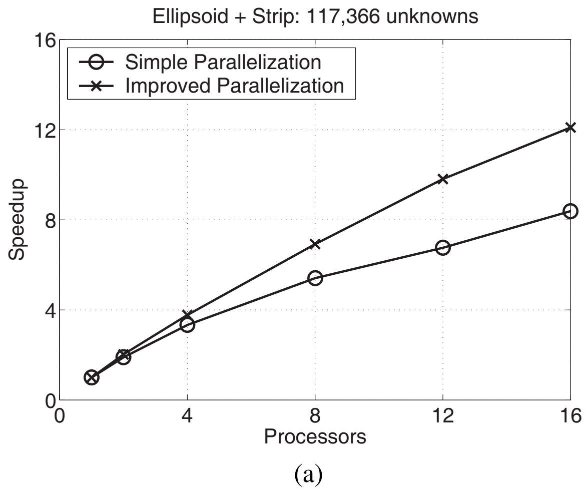
Fig. 2. (a) A geometry including a long strip over an ellipsoid, which is used in the tests of the parallelization of MLFMA. (b) Speedup obtained by the simple parallelization of MLFMA for the solutions of three different scattering problems, i.e., a sphere of radius 6λ, the geometry described in Fig. 2(a) at 500 MHz, and a long rectangular box with the dimensions of $\lambda \times 3\lambda \times 45\lambda$.

implementation using the shared-level concept proposed in [5]. The improved parallelization involves the application of different distribution schemes for the lower and upper levels of the tree structure. For the low levels of MLFMA, we still apply the simple strategy and distribute the clusters among the processors. Then, at an appropriate level (transition level), we switch to another strategy via an all-to-all communication, where the radiated and incoming fields are distributed among the processors. Above the transition level, each cluster is shared by all processors and each processor has the same angular portion of the radiated and incoming fields for all clusters.

Distributing the field instead of the clusters in the high levels of MLFMA has two consequences. First, the translations can be performed without any communication while the aggregation and disaggregation processes require one-to-one communications. It is favorable to avoid the dense communications of translations; however, the new communications in the aggregation and disaggregation steps should be handled carefully. Second, load-balancing is improved in the high levels of the tree structure since the distribution of the samples is easier than the distribution of clusters among the processors for these levels. It should be noted that the improved parallelization also includes two all-to-all communications in the transition level to pass between the two strategies applied for the upper and lower levels of the tree structure.

As an example, Fig. 3 compares the speedups of the simple and improved parallelizations for the scattering problems related to the geometry depicted in Fig. 2(a) at 500 MHz and the rectangular box with dimensions of $\lambda \times 3\lambda \times 45\lambda$. In both cases, the improved parallelization provides about 12 speedup for 16 processors. This corresponds to 75% efficiency, while the simple parallelization provides only 50% and 25% efficiency for the solutions of the same problems.

To formulate the scattering and radiation problems with open geometries, CFIE cannot be used and EFIE becomes the inevitable choice. However, EFIE produces ill-conditioned matrix



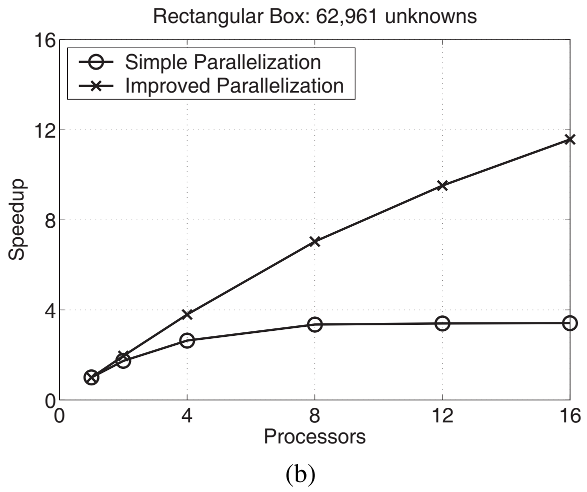


Fig. 3. Comparisons of the speedups obtained by simple and improved parallelizations of MLFMA in the solutions of the scattering problems involving (a) the geometry in Fig. 2(a) at 500 MHz and (b) the rectangular box with dimensions of $\lambda \times 3\lambda \times 45\lambda$.

equations that are difficult to solve iteratively. In addition, simple preconditioners do not improve the convergence of EFIE solutions. Therefore, the solution of open and large problems becomes a challenging task. The next section presents our efforts on the preconditioning of ill-conditioned EFIE systems and also further improving the iterative convergence of the CFIE systems via strong and efficient preconditioners.

4. Parallel Preconditioners for Large-Scale Problems

Simple preconditioners that use only diagonals or diagonal blocks of the near-field matrices do not provide sufficient reduction in the iteration counts for the ill-conditioned EFIE matrices. For the CFIE matrices, even though convergence can be achieved with easily parallelized BDP, further reduction in the solution time can be achieved with stronger preconditioners. Omitting some of the entries of the near-field matrix, particularly without considering the numeric values, degrades the performance of the preconditioners significantly. Hence, for large systems, we need preconditioners that use all or at least most of the information provided by the elements of the near-field matrices. However, neither the generation, nor the application of the preconditioner should exceed the $\mathcal{O}(N \log N)$ complexity of MLFMA.

Among such preconditioners, the commonly used incomplete factorization methods are based on eliminating some of the entries during the LU factorization [6]. After decomposing the near-field matrix in the form of $\overline{\boldsymbol{Z}}^{NF} \approx \overline{\boldsymbol{L}} \cdot \overline{\boldsymbol{U}}$, preconditioning operation is performed in each step by solving $(\overline{\boldsymbol{L}} \cdot \overline{\boldsymbol{U}}) \cdot \boldsymbol{v} = \boldsymbol{w}$, where $\overline{\boldsymbol{L}}$ and $\overline{\boldsymbol{U}}$ are the incomplete factors. A sparse approximate inverse (SAI) $\overline{\boldsymbol{M}} \approx (\overline{\boldsymbol{Z}}^{NF})^{-1}$, on the other hand, directly approximates the inverse of the matrix and application of the preconditioner is performed simply with the sparse matrix-vector multiplication $\boldsymbol{v} = \overline{\boldsymbol{M}} \cdot \boldsymbol{w}$. The backward and forward substitutions required in the incomplete factorization methods are inherently sequential and it is difficult to parallelize the preconditioner without sacrificing performance. On the other hand, sparse matrix-vector multiplications can be efficiently parallelized; hence approximate inverse type preconditioners are commonly preferred in parallel applications.

For approximate sparse inverse preconditioners, the approximation of the inverse of the near-field matrix is performed by minimizing

$$\|\overline{\boldsymbol{I}} - \overline{\boldsymbol{M}} \cdot \overline{\boldsymbol{Z}}^{NF}\|_F. \tag{4}$$

The approximation is done by enforcing $\overline{\boldsymbol{M}}$ to be sparse. With the Frobenius norm choice, the minimization can be performed independently for each row by using the identity

$$\|\overline{\boldsymbol{I}} - \overline{\boldsymbol{M}} \cdot \overline{\boldsymbol{Z}}^{NF}\|_F^2 = \sum_{i=1}^{n} \|\boldsymbol{e}_i - \boldsymbol{m}_i \cdot \overline{\boldsymbol{Z}}^{NF}\|_2^2, \tag{5}$$

where $\boldsymbol{e}_i$ is the ith unit row vector and $\boldsymbol{m}_i$ is the ith row of the preconditioner. The nonzero structure of the near-field matrix itself is a natural candidate for the nonzero pattern of SAI. Moreover, as noted by the work in [7], using the block structure of the near-field matrix, QR factorization involved in the least squares solutions of (5) can be done performed for each diagonal block. In this way, the construction time of the preconditioner can be reduced substantially.

Despite the advantages provided by SAI, we recognize that they do not provide close approximations to the near-field matrices for EFIE. Hence, we propose to use SAI for the iterative solution of the near-field system and then the near-field solution is used as a preconditioner to the original system [8]. For this purpose, a flexible solver is used for the original system, which allows preconditioning operation to change from iteration to iteration. This nested-solvers scheme yields better preconditioners compared to using SAI alone. On the other hand, for even larger problem sizes, where the near-field matrices become increasingly sparser, we need further information than that provided by the near-field matrix. Since the preconditioning operation can be performed iteratively with flexible solvers, MLFMA can also be considered for the preconditioning. We perform the matrix-vector multiplications faster by lowering the sampling densities of MLFMA. This way, the cost of the this effective preconditioning operation is significantly reduced.

As effective preconditioners, we compare BDP, SAI, and the iterative preconditioner obtained with an incomplete MLFMA, in which the inner system solution is also preconditioned with SAI. We denote this preconditioner with IFMM/SAI. For EFIE, we compare SAI, a preconditioner obtained from the iterative solution of the near-field system, which we name NF/SAI, and IFMM/SAI. The iterations are carried out on an 8-way SMP server with dual-core AMD Opteron processors. GMRES and FGMRES are used as the solvers and the convergence criterion is determined as a six order decay in the initial residual. The criterion for the inner system solution of IFMM/SAI is fixed as only a one order of decay in the initial residual or a maximum of 10 iterations. For NF/SAI, the stopping criteria for inner system solution is fixed as one order of decay in the initial residual or a maximum of 5 iterations. The experimental results reveal that such loose convergence criteria produce very effective preconditioners.

In Fig. 4, we show the solution times of the CFIE matrices for large problems, where the numbers of unknowns exceed 100,000, and for very large problems, where the unknowns

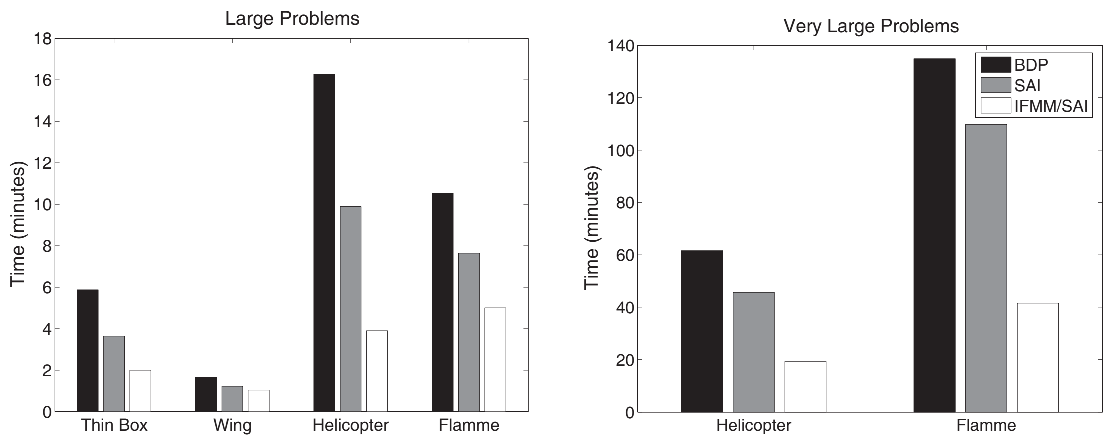


Fig. 4. CFIE results for "large problems" and for "very large problems." In the "large problem" category, the thin box has 147,800 unknowns, the wing has 117,945 unknowns, the helicopter has 183,546 unknowns, and the Flamme has 197,892 unknowns. In the "very large problem" category, the helicopter has 739,404 unknowns and the Flamme has 895,407 unknowns.

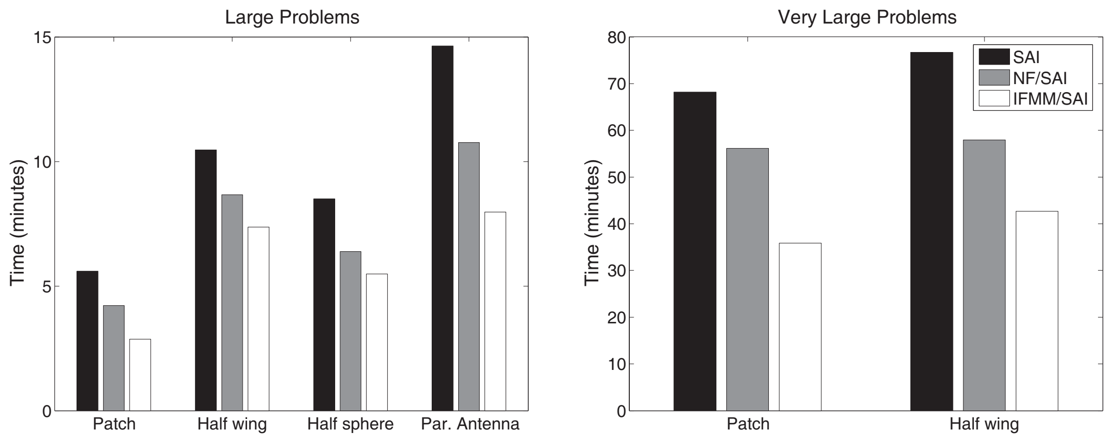


Fig. 5. EFIE results for "large problems" and for "very large problems." In the "large problem" category, the patch has 137,792 unknowns, the half wing has 127,925 unknowns, the half sphere has 116,596 unknowns, and the parabolic antenna has 356,439 unknowns. In the "very large problem" category, the patch has 719,000 unknowns and the half wing has 409,514 unknowns.

exceed 700,000. For both problem classes, SAI reduces the solution time by about $20\% - 30\%$, and IFMM/SAI reduces the solution time by a factor of 3, compared to the commonly used BDP. Using effective preconditioning strategies such as IFMM/SAI, we achieve the solution of a complex geometry such as a stealth target named Flamme [9] with approximately 1,000,000 unknowns in about 40 minutes. Moreover, IFMM/SAI have negligible memory requirement for storing the preconditioned Krylov subspace vectors [10].

In Fig. 5, we present the results related to highly insolvable EFIE matrices. We note that convergence cannot be attained in these problems using simple preconditioners such as BDP. SAI causes all iterations to converge in reasonable times. When we use the information carried by the near-field matrices more efficiently with NF/SAI, we reduce the solution times significantly, and when we use the incomplete MLFMA as a preconditioner, the solutions are achieved approximately two times faster compared to SAI.

5. Conclusion

Solution of large problems requires a focused effort to orchestrate numerous elements from different disciplines. Among those, integral-equation methods, discretization schemes, choice of basis and testing functions, iterative solvers, preconditioners, parallel architectures, parallelization strategies, and fast algorithms can be mentioned. In this work, we show that large problems involving millions of unknowns can be solved by developing improved standards in each one of those elements.

References

[1] A. J. Poggio and E. K. Miller, “Integral equation solutions of three-dimensional scattering problems,” in *Computer Techniques for Electromagnetics*, R. Mittra, Ed. Oxford: Permagon Press, 1973, Chap. 4.

[2] J. R. Mautz and R. F. Harrington, “H-field, E-field, and combined field solutions for conducting bodies of revolution,” *AEÜ*, vol. 32, no. 4, pp. 157–164, Apr. 1978.

[3] L. Gürel and Ö. Ergül, “Comparisons of FMM implementations employing different formulations and iterative solvers,” in *Proc. IEEE Antennas and Propagation Soc. Int. Symp.*, vol. 1, 2003, pp. 19–22.

[4] C.-C. Lu and W. C. Chew, “Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects,” *IEEE Trans. Antennas Propagat.*, vol. 45, no. 10, pp. 1488–1493, Oct. 1997.

[5] S. Velamparambil and W. C. Chew, “Analysis and performance of a distributed memory multilevel fast multipole algorithm,” *IEEE Trans. Antennas Propag.*, vol. 53, pp. 2719–2727, Aug. 2005.

[6] M. Benzi, “Preconditioning techniques for large linear systems: a survey,” *J. Comput. Phys.*, vol. 182, no. 2, pp. 418–477, July 2002.

[7] B. Carpentieri, I. S. Duff, and L. Giraud, “Experiments with sparse preconditioning of dense problems from electromagnetic applications,” CERFACS, Toulouse, France, Tech. Rep. TR/PA/00/04, 1999.

[8] T. Malas and L. Gürel, “Effective preconditioning techniques for iterative solutions of integral equation methods,” *IVth International Workshop on Electromagnetic Wave Scattering*, Gebze, İstanbul, Sept. 2006.

[9] L. Gürel, H. Bağcı, J. C. Castelli, A. Cheraly, and F. Tardivel, “Validation through comparison: measurement and calculation of the bistatic radar cross section (BRCS) of a stealth target,” *Radio Science*, vol. 38, no. 3, June 2003.

[10] Y. Saad, *Iterative Methods for Sparse Linear Systems.* Philadelphia: SIAM, 2003.