Effective Preconditioning Techniques for Iterative Solutions of Integral Equation Methods †

Tahir Malas[1] and Levent Gürel[1, 2]

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

Abstract

For iterative solutions of the integral equations employing MLFMA, we explore several preconditioning strategies. For sequential programs, we conclude that ILUT (incomplete LU with threshold) and ILU(0) perform superior for the electric-field integral equation (EFIE) and the combined-field integral equation (CFIE), respectively. For parallel implementations, we show that the sparse approximate inverse (SAI) preconditioner is quite successful for both EFIE and CFIE systems. However, for EFIE, SAI does not produce a very successful approximation to the near-field system. That is why we use SAI-preconditioned iterative solution of the near-field system as a more efficient preconditioner. On the other hand, for very large systems, neither of these methods succeeds to reduce the iteration counts as desired. The main reason of this failure is the thinning of the near-field matrices for increasing problem sizes. Considering this fact, we implement a preconditioner by solving an approximate system with an incomplete MLFMA. The superiority of this preconditioner is illustrated by solving a variety of large-scale problems.

1. Introduction

Integral-equation methods are commonly used to solve scattering and radiation problems in computational electromagnetics (CEM) because of their applicability to complex geometries and their high accuracy. Among others, these problems have two important applications. First, radar cross section (RCS) computations of arbitrarily shaped three-dimensional (3-D) targets are cast in the form of scattering problems, where the solution provides the unknown current density on the surface of the scatterer. Then, RCS can be easily computed from the surface current. Secondly, various antennae designs can also be performed accurately with these methods by solving the radiation problem.

The unknown current density on the target object is discretized by the method of moments (MOM), which yields complex, large, and dense linear systems. Hence, this method is limited by speed and storage capacity of the computer. However, with the help of the multilevel fast multipole algorithm (MLFMA), iterative methods can be used with $O(n \log n)$ computational complexity for an $n \times n$ system, provided that the number of iterations stays limited as the number of unknowns grows. In this way, large problems can be handled.

In each step, iterative methods require matrix-vector products with the coefficient matrix $\overline{\mathbf{Z}}$. MLFMA provides this operation to the solver by computing the near-field entries exactly and the far-field entries approximately but with controllable error. Hence, only the near-field part of the coefficient matrix $\overline{\mathbf{Z}}^{\mathrm{NF}}$ is stored and the interactions related to the far-field entries are carried out "on the fly." For this purpose, the computational domain is located in a cube, and then the cube is divided recursively into smaller cubes. This partitioning of the computational domain defines the levels of MLFMA. The near-field part corresponds to the interactions of the smallest cubes with their neighbors and their self-interactions. Since the smallest cube size is fixed in general to 0.25λ, a constant number of nonzero elements per row are stored in the sparse near-field matrix.

Two formulations of integral equations are widely used for the radiation and scattering problems. The electric-field integral equation (EFIE) is used for the geometries involving open surfaces, such as antennae. On the other hand, RCS computations usually involve targets with closed surfaces and the

† 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.

combined-field integral equation (CFIE) is the appropriate formulation for them since it is free from the internal resonance problem that EFIE may suffer. Moreover, CFIE yields linear systems that are easier to solve iteratively compared to EFIE.

However, neither EFIE nor CFIE matrices possess the properties of being positive definite or diagonally dominant, which facilitate convergence with the iterative solvers. In Figure 1, we show the pseudospectra of the EFIE and CFIE matrices for the 930-unknown sphere geometry. If the eigenvalues were clustered around the point (1,0), we would expect the iterative solvers to converge to the solution quickly. However, for the EFIE system, the eigenvalues are scattered along the imaginary axis and around the origin, which is an unfavorable situation. For the CFIE system, even though the eigenvalues are fairly clustered, a preprocessing step that moves the eigenvalues toward the point (1,0) can significantly accelerate the convergence.

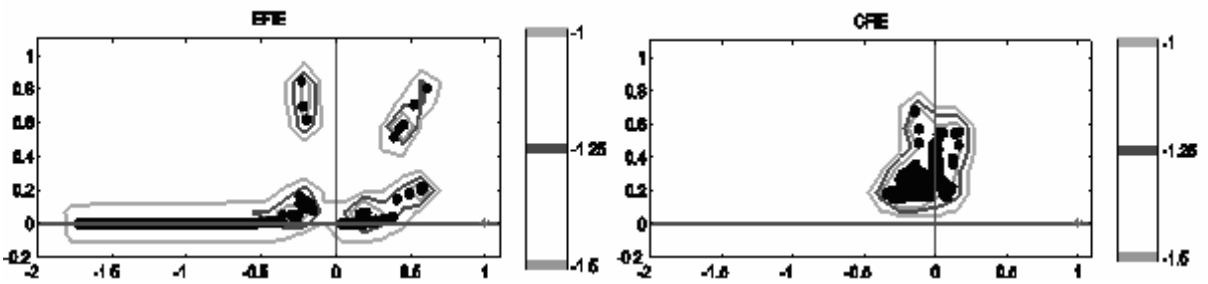

Figure 1. Pseudospectra of EFIE and CFIE matrices at the levels of 10^{-1}, $10^{-1.25}$, and $10^{-1.5}$. The black dots are the eigenvalues.

The preprocessing step to speed up the convergence of iterative solvers is called *preconditioning*. A *preconditioner* $\overline{\mathbf{M}}$ is a matrix or another operator that approximates the system matrix or its inverse. In forward type preconditioning, $\overline{\mathbf{M}}$ approximates the system matrix and we solve for $\overline{\mathbf{M}}^{-1} \cdot \overline{\mathbf{Z}} \cdot \mathbf{a} = \overline{\mathbf{M}}^{-1} \cdot \mathbf{v}$ (left-preconditioning) or $(\overline{\mathbf{Z}} \cdot \overline{\mathbf{M}}^{-1}) \cdot (\overline{\mathbf{M}} \cdot \mathbf{a}) = \mathbf{v}$ (right-preconditioning) instead of $\overline{\mathbf{Z}} \cdot \mathbf{a} = \mathbf{v}$. In this case, given a vector $\mathbf{y}$, it should not be expensive to solve for the system $\overline{\mathbf{M}} \cdot \mathbf{x} = \mathbf{y}$. In inverse type preconditioning, $\overline{\mathbf{M}}$ directly approximates the inverse of the system matrix. The application of the preconditioner is performed with the sparse matrix-vector multiplication $\mathbf{x} = \overline{\mathbf{M}} \cdot \mathbf{y}$, in which the approximate inverse $\overline{\mathbf{M}}$ should be sparse for efficiency.

In this paper, we summarize our efforts for the preconditioning of integral equations of CEM problems. First, we consider sequential implementations of MLFMA and concentrate on the incomplete LU (ILU) preconditioners, which are built from the near-field matrices. With some improvements for increasing their robustness, ILU preconditioners become very successful for both EFIE and CFIE formulations. However, because of the difficulty of the parallelization of the ILU preconditioners, we construct an efficient sparse approximate inverse (SAI) preconditioner for the parallel implementation of MLFMA. For EFIE, SAI is not as successful as ILU in reducing the iteration counts. Hence for a possible improvement, we use it as a preconditioner for the solution of the near-field system. Then, the solution of the near-field system is used as the preconditioner of the original system. For very large problems, on the other hand, due to their increasing sparseness, the near-field matrices do not serve as good approximations to the dense MOM matrices. Therefore, we use incomplete MLFMA to obtain a more powerful preconditioner. With these efforts, we are able to solve large-scale problems using a 16-processor computer in moderate solution times. The next three sections will detail the aforementioned preconditioning methods. Numerical results are presented in Section 5. Finally, we conclude with Section 6, summarizing the numerical results and providing some suggestions to MLFMA users.

2. Incomplete LU Preconditioners

Since MLFMA stores only the near-field matrix $\overline{\mathbf{Z}}^{\mathrm{NF}}$, we can use it as a preconditioner and solve (for example) the left-preconditioned system $(\overline{\mathbf{Z}}^{\mathrm{NF}})^{-1} \cdot \overline{\mathbf{Z}} \cdot \mathbf{a} = (\overline{\mathbf{Z}}^{\mathrm{NF}})^{-1} \cdot \mathbf{v}$. The inversion of the near-field

matrix can be performed via an LU factorization. However, during the factorization of sparse matrices, except for some specific cases, sparsity is lost. Nonetheless, by sacrificing some of the nonzero elements of the exact factorization, we end up with an incomplete LU (ILU) preconditioner. This idea is widely used and proven to be successful in solving sparse linear systems [2].

Two kinds of ILU preconditioners are developed depending on the criteria for the elimination of the matrix entries. First, consider an incomplete factorization of the near-field matrix, $\overline{\mathbf{Z}}^{\mathrm{NF}} = \overline{\mathbf{L}} \cdot \overline{\mathbf{U}}$. If we retain the nonzero values of $\overline{\mathbf{L}}$ and $\overline{\mathbf{U}}$ only at the nonzero positions of $\overline{\mathbf{Z}}^{\mathrm{NF}}$, we end up with the no-fill ILU method, or ILU(0). For problems that are not far from being diagonally dominant, such as the ones resulting from CFIE, this simple idea usually works well. On the other hand, since ILU(0) does not consider the numerical values of the entries, it becomes ineffective in predicting the locations of the largest entries for particularly nondiagonal dominant and highly indefinite matrices.

Alternatively, a second class of ILU preconditioners is developed that is based on the principle of dropping the matrix elements depending on their magnitudes. Among such methods, ILUT(τ , p) has been successful for systems obtained from a wide range of applications. During the factorization, ILUT drops matrix elements that are smaller than τ times the 2-norm of the current row; and of all the remaining entries no more than the p largest ones are kept. ILUT is known to yield more accurate and stable factorizations than level-of-fill methods with the same amount of fill-in [3]. However, for some cases, although the factorization terminates normally, the incomplete factors sometimes turn out to be unstable. When the problem is related to the small pivots, this problem can be avoided using partial pivoting as in the complete factorization case. The resulting preconditioner is called ILUTP [5].

3. Sparse Approximate Inverse Preconditioners

ILU preconditioners approximate the original matrix in the form of incomplete factors. Preconditioning operation is performed by backward and forward solves in each iteration. The main disadvantage of this preconditioner is the lack of parallelizability of the factorization and the application phases. To overcome this limitation, the SAI preconditioners, which are based on approximating the inverse of the matrix directly, are developed. For this purpose, an approximate inverse $\overline{\mathbf{M}} \approx (\overline{\mathbf{Z}}^{\mathrm{NF}})^{-1}$ is explicitly constructed and stored. Then, application of the preconditioner is carried out by a sparse matrix-vector multiplication, whose parallelization can be performed efficiently [2].

In this work, we concentrate on the SAI preconditioners depending on Frobenius norm minimization. For this class of preconditioners, the approximation of the inverse of the near-field matrix is performed by minimizing $\left\| \overline{\mathbf{I}} - \overline{\mathbf{M}} \cdot \overline{\mathbf{Z}}^{\mathrm{NF}} \right\|_{\mathrm{F}}$. The approximation is done by enforcing $\overline{\mathbf{M}}$ to be sparse. With the Frobenius norm choice, the minimization can be performed independently for each row by using the identity $\left\| \overline{\mathbf{I}} - \overline{\mathbf{M}} \cdot \overline{\mathbf{Z}}^{\mathrm{NF}} \right\|_{\mathrm{F}}^2 = \sum_{i=1}^{n} \left\| \mathbf{e}_i - \mathbf{m}_i \cdot \overline{\mathbf{Z}}^{\mathrm{NF}} \right\|_2^2$, where $\mathbf{e}_i$ is the ith unit row vector and $\mathbf{m}_i$ is the ith row of the preconditioner. Then, minimization is performed individually for each row of the preconditioner. Even though some communication is required at this stage, using proper data structures and efficient algorithms, high degree of scalability can be achieved.

The SAI preconditioners are in general not as successful as the ILU preconditioners. For this purpose, we considered using the SAI preconditioner for the iterative solution of the near-field system, and then the near-field solution is used as a preconditioner to the original system. This scheme yields a forward type preconditioner such as ILU preconditioner. The difference is that, in ILU preconditioning, the preconditioner is already in factorized form, i.e., $\overline{\mathbf{M}} = \overline{\mathbf{L}} \cdot \overline{\mathbf{U}} \approx \overline{\mathbf{Z}}^{\mathrm{NF}}$, and for a given vector $\mathbf{y}$, the system $\overline{\mathbf{M}} \cdot \mathbf{x} = \mathbf{y}$ is solved by using backward and forward solves. On the other hand, for the preconditioning scheme described, which we name as the iterative near-field preconditioner, the preconditioner is the exact near-field matrix, i.e., $\overline{\mathbf{M}} = \overline{\mathbf{Z}}^{\mathrm{NF}}$, but we approximately solve the system $\overline{\mathbf{M}} \cdot \mathbf{x} = \mathbf{y}$ by an iterative method.

For the iterative near-field preconditioner, the preconditioning operation is not fixed. Hence, a flexible solver should be used for the MOM matrix, allowing the preconditining operation change from iteration to iteration. We use FGMRES [2] in our experiments. For the near-field system solution, we use GMRES because of its ability to reduce the residual error quickly.

4. Using MLFMA for Effective Preconditioning

When the size of the problems gets larger, the near-field matrices become increasingly sparser and beyond some level it becomes nearly impossible to achieve low iteration counts with the preconditioners obtained from the near-field matrices. Therefore, for effective preconditioning of very large problems, we need more information than that provided by the near-field matrix.

This information can only be provided by MLFMA in the form of matrix-vector products. Hence, in each step of the iterative solver, we solve a nearby system using MLFMA. This preconditioning scheme yields a nesting of the solvers, which is shown in Figure 2.

Outer Solver: FGMRES, solve $\overline{\mathbf{Z}} \cdot \mathbf{a} = \mathbf{v}$
Mat-vec: MLFMA
Inner Solver: GMRES, solve $\overline{\mathbf{Z}}^{\text{Approx}} \cdot \mathbf{x} = \mathbf{y}$
Mat-vec: incomplete MLFMA
Preconditioner: SAI

Figure 2. Nesting solvers for preconditioning with MLFMA.

Since the inner solver is used for preconditioning purpose, we need to solve only an approximate system; hence there is no need to use a high-accuracy MLFMA for the matrix-vector operation. In order to perform the matrix-vector multiplication faster, we lower the sampling densities of MLFMA compared to the high-accuracy MLFMA. In this way, it is possible to reduce the cost of the preconditioning operation significantly.

5. Numerical Results

To demonstrate the effectiveness of the preconditioners described in Sections 2, 3, and 4, we perform sequential and parallel runs on 2-way and 16-way Opteron systems with 2 GB of memory per CPU. We use GMRES solver with no-restart because of its robustness. Starting with the zero initial guess, iterations are stopped when we reach 10^{-6} relative residual error or the maximum number of 1500 iterations. In the following tables, all times are given in terms of seconds.

For comparison purposes, we implement a sequential SAI and the following preconditioners:

- **LU:** Denotes the exact solution of the near-field matrix, which is used as a benchmark preconditioner. This solution is performed on another 64-bit server with 24 GB of memory. Due to its excessive computational requirements, it is presented merely for comparison of iteration numbers.
- **Block-Jacobi (BJ):** The blocks in this widely used preconditioner correspond to self-interactions of the smallest clusters. Exact inverse of each small block is obtained by LU decomposition. For an $n \times n$ system, there are n such blocks with fixed sizes and each of them contains a constant number of interactions. Hence, the cost of this preconditioner is $O(n)$.
- **Jacobi:** Contains only the diagonal of the near-field matrix. In Table 2, this preconditioner is included instead of BJ, since the latter performs poorer for EFIE.

Problem	Number of unknowns	LU	No PC		BJ		ILU(0)			SAI		
		Iter	Iter	Time	Iter	Time	Iter	Setup	Time	Iter	Setup	Time
Sphere	132,003	29	49	1,103	32	684	29	23	665	29	23,102	23,721
Thin box	147,180	37	158	1,965	106	1,290	45	271	1,025	64	298,478	299,301
Wing	117,945	31	86	1,110	52	779	32	46	542	37	73,100	73,586
Flamme	78,030	63	229	2,138	115	1,096	66	43	694	76	96,369	96,544
Helicopter	183,546	42	253	7,338	106	3,081	44	145	1,739	61	234,614	236,463

Table 1. ILU results for CFIE.

For CFIE, we use ILU(0) instead of ILUT, because ILUT does not perform better, but it has a higher setup time. From Table 1, we recognize that, especially for real-life problems such as Flamme [4] and helicopter, ILU(0) decreases the iteration counts and the solution times significantly. Moreover, the iteration counts turn out to be very close to that of LU, signaling the optimality of ILU(0). Finally, we note that SAI is not useful for CFIE in sequential programs due to its high setup time.

Problem	Number of unknowns	LU	Jacobi		ILUTP			SAI		
		Iter	Iter	Time	Iter	Setup	Time	Iter	Setup	Time
Patch	137,792	53	833	16,209	81	661	2,167	92	19,955	21,384
Open cube	171,655	332	-	-	376	2,243	9,833	354	207,436	213,619
Open prism	163,871	195	-	-	253	996	6,883	396	57,606	66,092
Half sphere	116,596	93	1,052	25,947	110	1,353	3,579	156	22,079	25,065

Table 2. ILU results for EFIE.

For open geometries in Table 2, first we note that effective preconditioning is indispensable for EFIE. Even with a robust solver, either the numbers of iterations are very high or no convergence is seen at all. ILUTP refers to ILUT preconditioner with pivoting, which is required for the robustness of ILUT. The parameters of ILUTP are chosen so that the memory requirement does not exceed that of the near-field matrix. From the table one can see that ILUTP reduces the iteration numbers by an order of magnitude compared to Jacobi when it succeeds to converge.

In Table 3, we compare SAI and BJ in a parallel environment for CFIE. We use larger problems for sphere, flamme, and helicopter. Even though the total times of SAI are higher than BJ, the iteration counts and the solution times are significantly lower. Hence, for the common multi right-hand-side (RHS) case, SAI is much more favorable. The last column in Table 3 denotes the number of RHS vectors required to compensate the relatively higher setup time of SAI.

CFIE		BJ				SAI				
Problem	Number of unknowns	Iter	Setup	Solution Time	Total Time	Iter	Setup	Solution Time	Total Time	RHS
Sphere	829,881	40	0.18	882	882	37	532	838	1,370	12
Thin box	147,180	104	0.14	352	352	63	304	218	522	3
Wing	117,945	53	0.05	98	98	38	192	73	264	8
Flamme	895,407	185	0.68	8,096	8,097	151	3,239	6,588	9,827	3
Helicopter	739,404	107	0.30	3,698	3,699	77	1,840	2,739	4,578	2

Table 3. Comparison of parallel SAI and BJ.

In Table 4, we compare LU, SAI, and iterative near-field (NF/SAI) preconditioners. For NF/SAI, we use SAI as a preconditioner for the iterative solution of the near-field system. For CFIE, the iteration counts of SAI are close to that of LU, but this is not the case for EFIE. However, with NF/SAI, we achieve very close iteration counts to that of LU. We note that, SAI preconditioned GMRES accelerates the convergence of the near-field system dramatically. We set the maximum number of iterations to five, and this suffices for an effective preconditioner.

Problem	Number of unknowns	LU	SAI Setup	SAI		NF/SAI	
		Iter		Iter	Solution Time	Iter	Solution Time
Patch	137,792	53	52	91	336	59	253
	719,000	MLE	214	190	4,091	141	3,369
Open prism	127,925	112	156	172	628	120	520
	409,514	MLE	336	389	4,601	209	3,476
Half sphere	9,911	38	7	60	24	40	17
	116,596	93	77	156	510	103	383
Reflector antenna	356,439	MLE	952	125	878	71	646

Table 4. Comparison of LU, SAI, and NF/SAI for EFIE. 'MLE' stands for 'memory limitation is exceeded.'

Since all these preconditioners are constructed from the near-field matrix, for very large problem sizes, they do not serve as effective preconditioners. In Figure 3, we present the total solution times of aforementioned parallel preconditioners and the preconditioner obtained from approximate MLFMA, for the largest problems shown in Tables 3 and 4. FMM/SAI is the preconditioner obtained with an incomplete MLFMA as explained in Section 4. As can be seen from Figure 3, using incomplete MLFMA for preconditioning purposes reduces the total solution time so significantly that even the solutions of large-scale problems can be obtained in less than an hour.

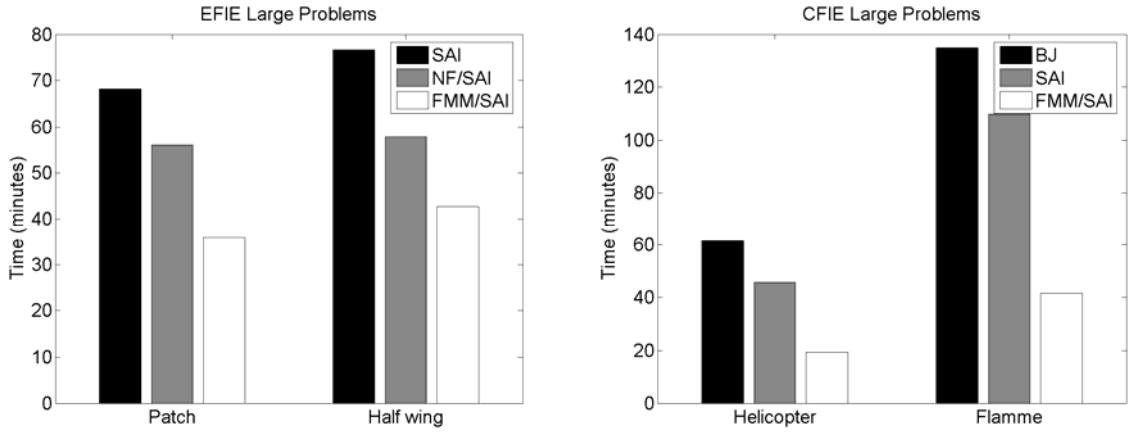

Figure 3. Comparison of MLFMA preconditioner with other parallel preconditioners.

6. Conclusion

In this work we show that, for the success of large-scale iterative solutions of integral equation methods, the key is effective preconditioning. Even with systems resulting from CFIE, when the size of the problem gets larger or the target geometry is complex as in real-life problems, preconditioning is required to achieve solutions with less time. On the other hand, severely ill-conditioned EFIE systems sometimes do not converge even with robust solvers such as the no-restart GMRES.

Our experiments with the sequential programs reveal that ILU preconditioners can be safely applied to integral equation methods, provided that pivoting is applied to ILUT for EFIE systems. The comparison with the exact solution of the near-field matrix shows the near optimality of them. Furthermore, ILU is the most established preconditioning technique, whose implementations are available in solver packages such as PETSc [5]. Hence, we strongly recommend their use for sequential problems.

For larger problems, on the other hand, effective parallelization of ILU is not possible; hence one should resort to SAI preconditioners. Though SAI works well up to certain problem sizes, more effective preconditioning strategies, such as the use of iterative solution of the near-field system or incomplete MLFMA help to solve even larger systems with low-memory and solution-time requirements.

References

[1] 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, 1997.

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

[3] E. Chow and Y. Saad, "Experimental study of ILU preconditioners for indefinite matrices," *J. Comput. Appl. Math.,* vol. 86, no. 2, pp. 387-414, 1997.

[4] L. Gürel, H. Bağcı, J.-C. Castelli, A. Cheraly, and F. Tardivel "Validation through comparison: Measurement and calculation of the bistatic radar cross section of a stealth target," *Radio Science,* vol. 38, no. 3, pp. 1046-1058, 2003.

[5] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, "PETSc users manual", Tech. Report ANL-95/11 Revision 2.1.5, Argonne National Laboratory, 2004.