A Two-level ILU preconditioner for electromagnetic applications[☆]

J. Cerdán, J. Marín, J. Mas

Instituto de Matemática Multidisciplinar,
Universitat Politècnica de València,
E-46022 Valencia. E-mail: jcerdan,jmarinma,jmasm@imm.upv.es

Abstract

Computational electromagnetics applications based on the solution of the integral form of Maxwell's equations with boundary element methods require the solution of large and dense linear systems. For large-scale problems the solution is obtained by using iterative Krylov-type methods provided that a fast method for performing matrix-vector products is available. In addition, for ill-conditioned problems some kind of preconditioning technique must be applied to the linear system in order to accelerate the convergence of the iterative method and improve its performance. For many applications it has been reported that incomplete factorizations often suffer from numerical instability due to the indefiniteness of the coefficient matrix. In this context, approximate inverse preconditioners based on Frobenious-norm minimization have emerged as a robust and highly parallel alternative. In this work we propose a two-level ILU preconditioner for the preconditioned GMRES method. The computation and application of the preconditioner is based on graph partitioning techniques. Numerical experiments are presented for different problems and show that with this technique it is possible to obtain robust ILU preconditioners that perform competitively compared with Frobenious-norm minimization preconditioners.

Keywords: Computational electromagnetism, Iterative methods, Preconditioning, Incomplete LU factorizations, Graph partitioning, Matrix

[☆]This work was supported by the Spanish Ministerio de Economía y Competitividad under grant MTM2014-58159-P.

reorderings
2010 MSC: 65F10, 65F08, 65F50, 31A10

1. Introduction

The numerical solution of the Maxwell's equations [1] plays a crucial role in numerous large scale industrial and scientific applications related with electromagnetism phenomena. To name a few, the computation of the antenna radiation pattern, electromagnetic interference and compatibility studies of an electrical device with their environment, and scattering problems as the computation of the radar cross-section of a 3D body are important for aerospace industry. The performance of a computational electromagnetism (CEM) code is associated with the strengths and weaknesses of underlying numerical methods chosen for its implementation. Overall, for real-life applications the computation of an approximate solution of the linear systems arising from the discretization of the Maxwell's equations is the most demanding part in terms of computer resources. Thus, devising efficient numerical algorithms for solving these linear systems is key to develop codes capable to run with a good performance in modern computer architectures.

The most common techniques for obtaining a numerical solution of Maxwell's equations can be classified either into methods that solve the differential equations or methods that consider their integral formulation. Partial differential equations methods (PDEMs) use classical techniques like the finite-element or the finite-difference method to discretize directly the Maxwell's equations [2, 3]. An advantage of these methods is that they allow the simulation of complex electrical structures. By contrast, the study of electrical phenomena in open domains is rather difficult and artificial boundary conditions must be imposed to simulate an infinite volume. With the rise of modern computer architectures and the sustained increment of the computational resources available, integral equations methods (IEMs) have emerged as an attractive alternative for CEM applications. These methods solve the problem by reformulating the Maxwell's

equations as a set of integral equations with equivalent sources [4, 5]. The integral equations relate the electric and magnetic fields to the equivalent electric and magnetic currents on the surface of the object. This leads to a reduction on the dimensionality of the problem by one, and therefore allows significant reduction on the number of unknowns of the associated linear systems. Because the boundary conditions are incorporated into the surface integral equations, IEMs can handle general geometries in open domains without formulating any artificial boundary. Thus, they are attractive for a wide range of industrial simulations in open geometries.

The integral equations are usually discretized by means of the boundary element method (BEM) or the Method of Moments (MoM) [6, 7]. Unlike PDEMs, the matrices arising from IEMs are dense and expensive to solve. Since in large-scale industrial applications the size of the matrices can be very large the application of direct gaussian elimination methods is out of context, leaving the use of Krylov-type iterative methods as the only practical alternative. The arithmetic complexity of these methods resides on the computation of matrix-vector products, operation that has a complexity of order $\mathcal{O}(n^2)$. This complexity can be reduced to $\mathcal{O}(n \log n)$ by applying optimized methods as the Fast Multipole Method [8]. An additional difficulty is that in many applications IEMs have to deal with ill-conditioned matrices that are challenging to solve, as it is the case of the matrices arising from the discretization of the electric field integral equations (EFIE).

As it is well known, the success of an iterative method for ill-conditioned problems depends on applying a suitable preconditioning technique to the system matrix. In the case of EFIE most algebraic factorized preconditioners fail to produce good converge rates or even fail to converge, see [9, 10, 11, 12]. The best results on medium size problems were obtained with sparse approximate inverse preconditioners based on Frobenious norm minimization [13]. Nevertheless, for large problems the relative nonzero density of the preconditioner is too small with a negative effect on the performance of this class of preconditioners. These problems may be overcome by performing spectral low-rank updates (SLRU) of

the preconditioned matrix [14, 15, 16]. This technique consists in shifting by one a subset of the smallest eigenvalues that play a key role in slowing down the convergence of Krylov methods. The results of the numerical experiments show that the SLRU technique can improve considerably the performance, specially when multiple right hand sides have to be solved as is the case for scattering problems. Alternative techniques implement flexible variants of the GMRES method [17].

Our aim in this work is to present a technique for the computation of ILU-type preconditioners for ill-conditioned CEM applications. The method is based on graph partitioning techniques applied to the near field matrix of the linear systems. The paper is organized as follows. In Sections 2 and 3 we review the main ideas of graph partitioning and the algorithm for computing a two-level ILU for CEM applications is presented. Then, the numerical results are presented in Section 4. Finally, the main conclusions are outlined in Section 5.

2. Graph partitioning

Graph partitioning is a widely used technique in parallel processing as it provides an effective way to distribute unstructured computations among processors. This decomposition is achieved by splitting the adjacency graph of a matrix into p parts subject to some constraints. Here we will describe how it can be used to compute a two-level ILU preconditioner. Although a number of different methods have been proposed in the literature [18], the idea behind graph partitioning is the computation of a $p-$way partitioning of the graph keeping the size of the p subgraphs balanced while minimizing to some extent the number of edges that are cut. Let us describe briefly the technique.

Let A be a sparse structurally symmetric matrix. The associated undirected adjacency graph $G = (V, E)$ consists of a set nodes $V = \{1, \ldots, n\}$, one node for each row or column of the matrix, and the edge set E. There is an edge $\langle i, j\rangle$ for any matrix entry $a_{ij} \neq 0$. Note that there is not distinction between $\langle i, j\rangle$ and $\langle j, i\rangle$. For nonsymmetric sparse patterns the adjacency graph of $A + A^T$

is considered instead. We define the separator set as the group of nodes which are connected by edges that are cut in the graph partition. We also define the group of interior nodes as the nodes which are connected with the separator set. Thus, there are p groups of interior nodes, one for each subgraph of the partition.

By numbering first the interior nodes and taking the separator set last, the matrix is permuted into the following block angular form:

$$P^T AP = \begin{pmatrix} A_1 & & & & B_1 \\ & A_2 & & & B_2 \\ & & \ddots & & \vdots \\ & & & A_p & B_p \\ C_1 & C_2 & \dots & C_p & A_S \end{pmatrix} \tag{1}$$

where P is a permutation matrix. The diagonal blocks $A_1, ..., A_p$ correspond to subgraphs induced by the interior nodes in the graph decomposition, the off-diagonal blocks B_i y C_i represent the connections between interior nodes and the separator set, and A_s correspond to the subgraph induced by the separator set. By computing an incomplete factorization for a matrix structured as in (1) one may obtain an efficient preconditioner for solving linear systems iteratively.

3. Two-level ILU preconditioner

Consider a linear system of n equations with n unknowns given by

$$Ax = b \tag{2}$$

obtained after the discretization of the integral form of the Maxwell's equations using the Method of Moments. The matrix A is called the impedance matrix and it is dense, non-hermitian and with complex elements. Moreover, the impedance matrix is often characterized by a large condition number which results in a slow convergence of iterative methods. A preconditioning technique consists in finding a matrix M for which an approximate solution of the equivalent linear

systems

$$M^{-1}Ax = M^{-1}b\,, \quad \text{or} \quad AM^{-1}y = b\,, \quad x = M^{-1}y \tag{3}$$

is obtained more efficiently. The matrix M is called the preconditioner. If the preconditioned matrix $M^{-1}A$ (or AM^{-1}) has a better condition number than A or its eigenvalues have a favorable distribution, one can expect an improvement of the convergence rate of the iterative method [19].

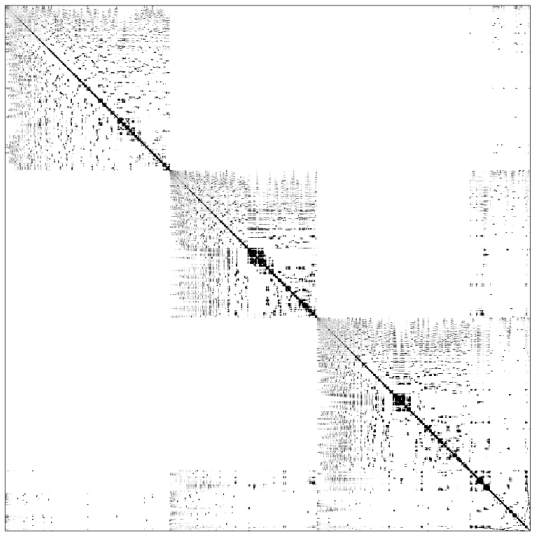

Figure 1: A 3−partitioning for CETAF10

The impedance matrix relates the induced currents with the incident fields on the surface of a 3D body. Each equation represents the interaction between an edge of the mesh and its neighborhood. In general, the magnitudes of its entries, associated with the electric and magnetic field operators, decrease with the distance between edges of the mesh. In addition, due to the rapid decay of the discrete Green's function, the number of entries with relative large magnitude compared to the others can be very small. Thus, it is possible to obtain a sparse approximation of A by considering only the near-field entries that significantly affect the spectral properties of the integral equation kernel. If we decompose the impedance matrix into its near-field and far-field entries, equation (2) can

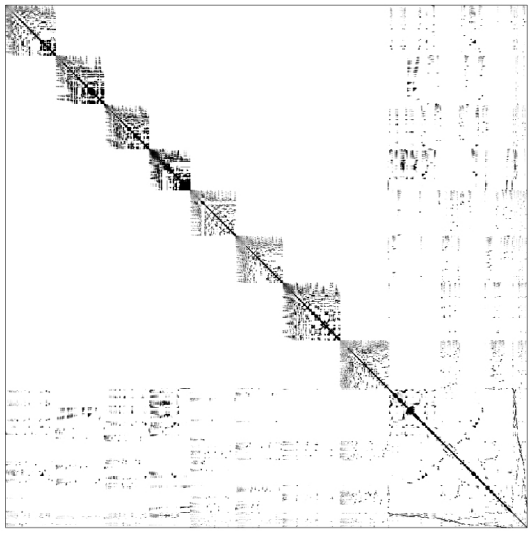

Figure 2: An 8−partitioning for CETAF10

be rewritten as

$$(A^{near} + A^{far})x = b \ , \tag{4}$$

where the near-field matrix A^{near} contains only those entries representing the interactions between source and test basis functions lying within some threshold distance.

A good preconditioner should approximate the inverse of A or, at least, its near-field entries. Therefore, the preconditioner will be formulated using the matrix A^{near}. After computing a $p-$way partitioning for the adjacency graph of A^{near} and permuting the matrix as described in the previous section, one obtains the block angular form (1). Figures 1 and 2 show the sparse pattern of two different permutations of a near-field matrix obtained from matrix CETAF10. Notify that, for a fixed graph, the number of nodes in the separator set increases with the number of partitions while the number of interior nodes in each partition decreases. As it will be discussed later, a larger size of the matrix A_S may lead to an increment of the preconditioner setup time. The block LU

factorization of a matrix with block angular form (1) is

$$P^T A^{near} P = \begin{pmatrix} L_1 & & & & \\ & L_2 & & & \\ & & \ddots & & \vdots \\ & & & L_p & \\ F_1 & F_2 & \dots & F_p & L_S \end{pmatrix} \begin{pmatrix} U_1 & & & & E_1 \\ & U_2 & & & E_2 \\ & & \ddots & & \vdots \\ & & & U_p & E_p \\ & & \dots & & U_S \end{pmatrix}, \quad (5)$$

where $A_i = L_i U_i$, $E_i = L_i^{-1} B_i$, $F_i = C_i U_i^{-1}$. The matrices L_S and U_S are the triangular factors of the Schur complement matrix

$$S = A_S - \sum_{i=1}^{p} C_i A_i^{-1} B_i. \quad (6)$$

From (5) an incomplete factorization $P^T A^{near} P$ is obtained by computing an ILU decomposition for each diagonal block, i.e., $A_i \approx \hat{L}_i \hat{U}_i$. Moreover, since $A_i^{-1} \approx U_i^{-1} L_i^{-1}$, it also follows from equation (6) that an approximation of the Schur complement matrix S is computed and factorized as $\hat{S} \approx \hat{L}_S \hat{U}_S$. This double factorization and approximation characterizes the two-level nature of the algorithm. It is worth to note that fill-in produced in the factorization process is located within the nonzero blocks of $\hat{L}$ and $\hat{U}$. Thus, the preconditioner $M = \hat{L}\hat{U}$ preserves a good deal of sparsity compared with the inverse of A^{near} that is generally full. Moreover, additional sparsity can be obtained by applying a fill-in reordering strategy to each diagonal block A_i as, for instance minimum degree, or a recursive application of graph partitioning [20]. The last option is of particular interest in the context of parallel computations. With respect to the computational complexity of the preconditioner computation, note that it depends heavily on the dimension of the Schur complement matrix $\hat{S}$ since it must be computed and factorized. Therefore, it is important to keep the separator set as small as possible. That is, for a fixed problem size there will be a maximum number of partitions that can be used efficiently.

The ILU factorization described is used as a preconditioner for the iterative solution of the permuted linear system

$$(P^T A P) y = P^T b, y = P^T x,$$

where A is the full matrix. Thus, each preconditioning step $Ms = r$ consists in two triangular solves. Assuming that the vectors are partitioned conformally to $P^T A^{near} P$, it is first computed the solution of the lower triangular system $\hat{L}y = r$ by forward substitution,

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \\ y_S \end{pmatrix} = \begin{pmatrix} \hat{L}_1^{-1} r_1 \\ \hat{L}_2^{-1} r_2 \\ \vdots \\ \hat{L}_p^{-1} r_p \\ \hat{L}_S^{-1} (r_S - \sum_{i=1}^{p} \hat{F}_i y_i) \end{pmatrix} \tag{7}$$

followed by the solution of the upper triangular system $\hat{U}s = y$.

$$\begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_p \\ s_S \end{pmatrix} = \begin{pmatrix} \hat{U}_1^{-1} (y_1 - \hat{E}_1 s_S) \\ \hat{U}_2^{-1} (y_2 - \hat{E}_2 s_S) \\ \vdots \\ \hat{U}_p^{-1} (y_p - \hat{E}_p s_S) \\ \hat{U}_S^{-1} y_S \end{pmatrix} \tag{8}$$

Note that it is not necessary to compute and store explicitly the off-diagonal blocks $\hat{E}_i$ and $\hat{F}_i$ since their application over a vector can be done with a matrix-vector product and a triangular solve.

Remark. As mentioned in the Introduction, it is important for large-scale problems to have at disposal highly parallel preconditioning algorithms. Although it is not considered in this work, we want to comment some aspects about the parallel computation and application of the preconditioner, besides its potential performance. Assuming that there are p computational nodes available, the permuted near-field matrix in angular form (1) can be distributed among them, each node storing a row-block and a column-block of the matrix, i.e., the node N_i stores the blocks A_i, B_i and C_i for $i = 1, \ldots, p$. With this distribution the ILU factorizations of the diagonal blocks A_i can be performed completely in parallel. The Schur complement matrix S given in (6) is obtained after a fan-in process with the final assembly taking place in one of the nodes, that also must

hold the diagonal block A_S. Then, the Schur complement matrix is factorized, being this task the sequential part of the preconditioner computation and a potential bottleneck. The preconditioning step is also highly parallel. Assume that the vectors are also distributed accordingly to the matrix. After a sequential computation of the part of the solution vector that corresponds to the Schur complement matrix and distributing this part among the computer nodes, the solution of the upper triangular system (8) can be done completely in parallel. The lower triangular system (7) requires a final fan-in across the processors. We refer to [21] where a similar technique was used to compute highly parallel factorized sparse approximate inverse preconditioners for large-scale PDE problems with excellent scalability.

4. Numerical experiments

In this section we show the results of the numerical experiments obtained for a set of model problems which are listed in Table 1. All the matrices were kindly provided to us by the EADS-CASA company. They correspond to the EFIE formulation of the Maxwell's equations for the CETAF problem with different sizes, and other private test problems of the company. All the problems arise from the computation of a radar cross section and are challenging to solve by an iterative method, specially the set CN which are highly indefinite matrices. The company provided us the discretized impedance matrix, the right hand side vector, a reference solution vector and a matrix with distances between the elements of the mesh. This matrix was used to filter the impedance matrix with a threshold of 0.04 meters in all the cases. The table shows the matrix dimension n, and the relative nonzero density of the near-field matrix used to compute the two-level ILU preconditioner, $\rho(A^{near})$.

All codes developed for the tests were written in FORTRAN 95 in double precision complex arithmetic, compiled with the Intel Fortran Composer XE 2013 and linked with the Intel Math Kernel Library. The codes were run in a Linux PC with an intel Core i5 processor and 8 Gb of RAM. The restarted

Matrix	n	$\rho(A^{near})$
CETAF3	3097	0.02
CETAF5	5021	0.02
CETAF10	10022	0.01
CN2	2038	0.02
CN3	3020	0.02
CN5	5005	0.02

Table 1: Tested matrices

GMRES(200) method with right preconditioning was used to solve each linear system [22]. That was equivalent in practice to the full GMRES since convergence was attained before restarting, with the exception of the matrices CETAF10 and CN2. The iterative method was stopped when the initial residual was reduced by at least six orders of magnitude which is more than enough to obtain accurate radar cross section results. The initial guess is set to the zero vector.

The METIS software package [23] was used to obtain a $p-$way partitioning of the near-field adjacency graph. METIS is a set of serial programs for partitioning graphs based on multilevel recursive-bisection and multilevel p-way partitioning schemes. It produces high quality partitions in a very short time relative to the overall solution time. Moreover, there exists also a parallel version specially suited for computations in parallel architectures. The ILUT algorithm was used to compute incomplete factorizations [19]. ILUT performs an ILU with threshold and also allows for restricting the maximum number of nonzeros by row. For simplicity, in our tests only the threshold option was used to reduce the fill-in which is indicated in the caption of the tables. Some internal experiments, not listed here, did not lead to any significant advantage that may justify to add an extra complexity to the analysis of the experimental results.

The numerical experiments were conducted to show the effect of the num-

ber of partitions on the quality of the ILU preconditioner, the effect of applying reorderings to the diagonal blocks of the permuted near-field matrix before computing the incomplete factorization, and finally a comparison of the proposed two-level ILU preconditioner with SPAI, an approximate inverse preconditioner which is widely used in CEM applications.

It is well known that incomplete factorizations for nonsymmetric matrices can benefit from symmetric reorderings applied to the coefficient matrix, such as fill-in reducing orderings and level set reorderings. While level set reorderings are in general better for incomplete factorizations (see [24]), minimum degree results in incomplete factors which are sparser. This can be a deciding factor in CEM applications that usually demand large amount of computer memory. Therefore, we tested the effect of the multiple minimum degree, quotient minimum degree, reverse Cuthill-McKee and nested dissection [25, 26]. In tables we refer to them as *mmd*, *qmd*, *rcm* and *nd*, respectively, whereas *no* indicates natural ordering. Moreover, as reported in [27], the robustness and performance of Krylov subspace methods preconditioned with incomplete factorizations for highly indefinite and nonsymmetric matrices can be improved by moving large entries to the diagonal of the matrix. This task can be accomplished with nonsymmetric reorderings such as the maximum sum transversal and the maximum product transversal algorithms [28].

Tables 2 and 3 show the effect of the symmetric reorderings for the matrices CETAF3 and CN3, respectively. For the rest of matrices similar behavior was observed. In this table, p indicates the number of partitions of the adjacency graph of A^{near}, *Symm* indicates the symmetric reordering method applied to the diagonal blocks and Schur complement matrix, *Tr* is the time spent to compute these reorderings, ρ is the density of the preconditioner with respect the number of nonzeros of A^{near}, *iter* is the number of iterations, *Tp* and *Tsol* are the preconditioner computation time and iterative solution time, respectively. All timings correspond to the CPU time in seconds.

First, we observe that sparser preconditioners are computed as result of permuting the near-field matrix and, at the same time, the number of iterations

needed to converge decreased considerably. And this was observed for any number of partitions of the adjacency graph. In fact, the application of the multiple minimum degree alone, i.e. $p = 1$, was enough to reach convergence for the set of matrices CN. In general we found that multiple minimum degree performed slightly better than the others since with less fill-in the convergence rate was on a par, although no big differences with quotient minimum degree and reverse Cuthill-Mckee were found. Nested dissection also produced good results but to a less extent. Moreover, the time needed to compute and apply a symmetric reordering is very small compared with the overall solution time. Thus, we strongly recommend the application of symmetric reorderings, specially a minimum degree type algorithm.

With respect to nonsymmetric reorderings, some representative results for the matrices CETAF3, CETAF5, CN3 and CN5 are shown in Tables 4 to 7. Unlike symmetric reorderings, we did not find big benefits applying this kind of reorderings for the matrices tested. The results were always pretty similar. Sometimes one of them performed slightly better than the others and sometimes the opposite, with no recognizable trend. Nevertheless, since this preprocessing may have a stabilizing effect on the computation of the preconditioner and its applications is inexpensive, it can be worth to test them in case of preconditioner computation failure or instabilities.

Let us analyze the two-level ILU preconditioner proposed. From Tables 2, 3 and 8 one observes immediately that the application of the graph partitioning technique first reduces the number of iterations, and second allows for the computation of sparser preconditioners. These observations can be made even without applying symmetric reorderings, although its application helps to obtain improved preconditioners. Moreover, a nice feature is that the number of iterations remain quite stable with the number of partitions. The approximate Schur complement contribution to the preconditioner that works as a coarse grid correction between the computational subdomains explains this effect. Since IEMs lead to a reduction on the dimensionality of the problem by one, the observed behaviour may be in line with [21], where specially good results for 2D PDE

problems were obtained with a two-level approximate inverse preconditioner.

Table 9 shows the results obtained with SPAI. A standard implementation of the SPAI preconditioner with an a priory sparsity pattern obtained from the near-field matrices filtered with a 0.6 meters threshold was used. With this threshold both, the two-level ILU and SPAI preconditioners, had similar nonzero densities. That is, the two-level ILU preconditioner and near-field matrix used to compute SPAI had a roughly equal amount of nonzeros. Sparser preconditioners lead to lower convergence rates for both preconditioners with restarted GMRES, situation that may be present for large-scale applications where the available computer memory may be a limiting factor. In those scenarios a combination of techniques based on spectral low-rank updates or flexible variants of the GMRES must be applied in order to regain satisfactory convergence rates. The study of these and other refined techniques are out of the scope of this paper. Thus, to fully understand the possibilities of the two-level preconditioner compared with SPAI we do not think that it is necessary to show the results for very sparse preconditioners. Nevertheless, the nonzero density with respect to the full impedance matrix A was about 2 to 4 percent, in line with studies that appear in the bibliography for similar size problems. The sparser preconditioners relative to the size of the problem were the corresponding ones used with the matrix CETAF10.

We found that for the matrices tested the two-level ILU preconditioner always converged faster both in time and in number of iterations. Taking into account that computing and incomplete LU is also considerably cheaper, we think that the two-level ILU preconditioner proposed is a competitive alternative for CEM applications. Indeed, we think that the increment of the preconditioner computation time with the size of the preconditioner is a limiting factor for SPAI preconditioners that is somehow relieved with the two-level ILU preconditioner, at least in sequential computations.

Table 10 and Figures 3 to 5 show a summary of the results obtained for the two-level preconditioner related to the Schur complement block size. The results correspond to the preconditioner computed with symmetric *mmd* ordering

and without nonsymmetric reordering applied. *Ratio* represents the quotient between the average diagonal block size and the Schur complement size. The iterations and total CPU time in seconds are also listed. One can observe the behavior on the number of iterations and time with respect to the number of partitions highlighted above. Finally, let us discuss the potential degree of parallelism of the algorithm using these figures. In Section 3 we pointed out some ideas about how to develope an efficient implementation of the two-level preconditioner. We observe that the Schur complement size increases with the number of partitions since, for a fixed size matrix, the number of cut edges increases. Since the computation of the ILU factorization of the Schur complement matrix is sequential, keeping a separator set as small as possible is important for the parallel efficiency of the algorithm. Based on the results from [21] the parallel efficiency of the algorithm should start to decrease when the Schur complement size surpasses the size of the diagonal blocks, that is, when the parameter *Ratio* becomes smaller than one. Above this value an improvement on the overall solution time and a good scalability of the algorithm may be expected.

5. Conclusions

We have presented a method for computing two-level incomplete LU factorizations for CEM problems that are solved by means of an integral equation method. The algorithm obtains a block LU factorization of the near-field matrix that has been previously permuted to block angular form. The permutation is based on finding a $p-$way partitioning of the adjacency graph of the matrix. The numerical experiments obtained for some EFIE test problems show that the two-level LU preconditioners computed with this strategy accelerate considerably the convergence rate of the GMRES method. It has been observed that the proposed algorithm not only reduces the number of iterations and time, but also allows for the computation of sparser preconditioners. Moreover, it has been shown that the combination of the two-level ILU preconditioner with symmetric reorderings, specially multiple minimum degree, helps to obtain robust

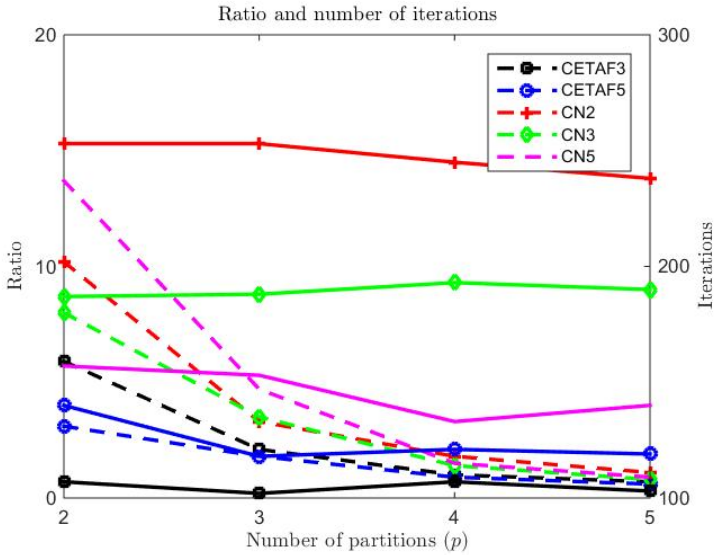

Figure 3: Average diagonal block size to Schur complement size ratio (dashed lines), and iterations count with respect to the number of partitions of the adjacency graph for the matrices tested except CETAF10.

preconditioners while at the same time the amount of fill-in is reduced.

References

[1] J. C. Maxwell, A dynamical theory of the electromagnetic field, Philosophical Transactions of the Royal Society 155 (1865) 459–512.

[2] K. Kunz, R. Luebbers., The finite difference time domain method for electromagnetics., SIAM J. Sci. Comput. 18 (3) (1997) 838–853.

[3] P. P. Silvester, R. L. Ferrari., Finite Elements for Electrical Engineers, Cambridge University Press, Cambridge, 1990.

[4] W. C. Chew, J.-M. Jin, E. Michielssen, J. Song, Fast and Efficient Algorithms in Computational Electromagnetics and Elastic Waves, Morgan & Claypool Publishers, 2009.

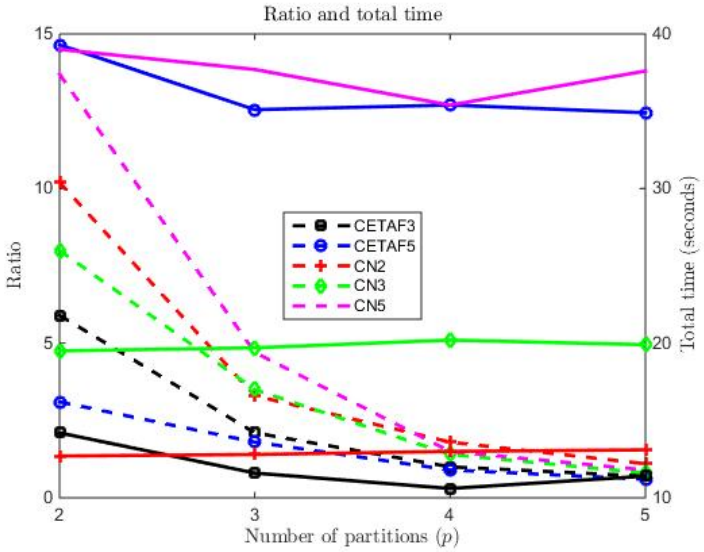

Figure 4: Average diagonal block size to Schur complement size ratio (dashed lines), and total solution CPU time in seconds with respect to the number of partitions of the adjacency graph for the matrices tested except CETAF10.

[5] W. Hackbusch, Integral Equations, Vol. 120 of International Series of Numerical Mathematics, Birkhauser Basel, 1995. doi:10.1007/978-3-0348-9215-5.
URL http://dx.doi.org/10.1007/978-3-0348-9215-5

[6] R. Harrington., Origin and development of the method of moments for field computation., IEEE Antennas and Propagation Magazine 32 (1990) 31–35. doi:10.1109/74.80522.

[7] S. M. Rao, D. R. Wilton, A. W. Glisson., Electromagnetic scattering by surfaces of arbitrary shape, IEEE Trans. Antennas Propagat. AP-30 (1982) 409–418.

[8] L. Greengard, V. Rokhlin., A fast algorithm for particle simulations, J. of Computational Physics 73 (3) (1987) 325–348.

[9] B. Carpentieri., Sparse preconditioners for dense linear systems, from elec-

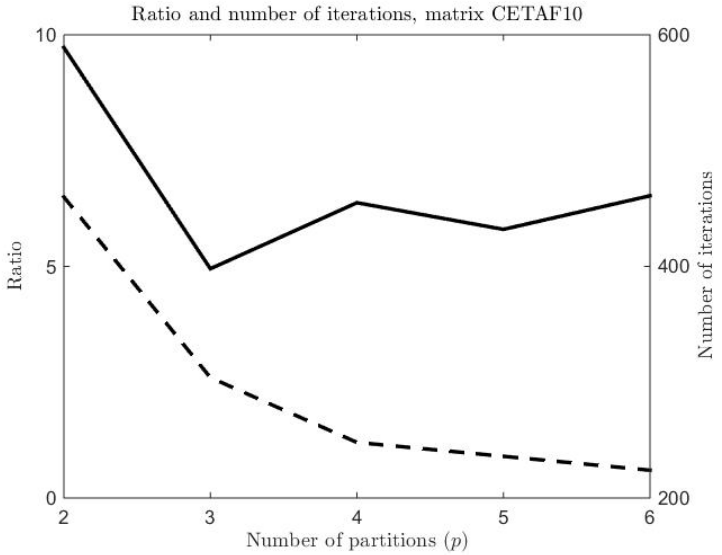

Figure 5: Average diagonal block size to Schur complement size ratio (dashed lines), and iterations count with respect to the number of partitions of the adjacency graph for CETAF10.

tromagnetic applications, Ph.D. thesis, PhD Thesis, l'Institut National Polytechnique de Toulouse, CERFACS (2002).

[10] B. Carpentieri, I. S. Duff, L. Giraud, Sparse pattern selection strategies for robust Frobenius-norm minimization preconditioners in electromagnetism., Numerical Linear Algebra with Applications 7 (2000) 667–685.

[11] B. Carpentieri, I. S. Duff, L. Giraud, M. Magolu Monga Made, Sparse symmetric preconditioners for dense linear systems in electromagnetism, Numerical Linear Algebra with Applications 11 (2004) 753–771.

[12] B. Carpentieri, I. S. Duff, L. Giraud, G. Sylvand, Combining fast multipole techniques and an approximate inverse preconditioner for large electromagnetism calculations, SIAM J. on Scientific Computing 27 (3) (2005) 774–792.

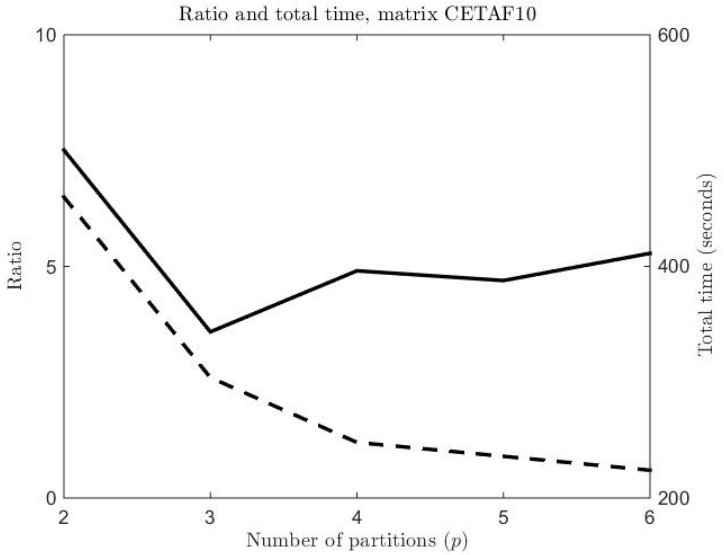

Figure 6: Average diagonal block size to Schur complement size ratio (dashed lines), and total solution CPU time in seconds with respect to the number of partitions of the adjacency graph for CETAF10.

[13] M. J. Grote, T. Huckle, Parallel preconditioning with sparse approximate inverses, SIAM J. on Scientific Computing 18 (3) (1997) 838–853.

[14] B. Carpentieri, I. S. Duff, L. Giraud, A class of spectral two-level preconditioners, SIAM J. on Scientific Computing 25 (2) (2003) 749–765.

[15] B. Carpentieri, L. Giraud, S. Gratton, Additive and multiplicative two-level spectral preconditioning for general linear systems, SIAM J. on Scientific Computing 29 (4) (2007) 1593–1612 (electronic).

[16] J. Mas, J. Cerdán, M. Malla, J. Marín, Application o the jacobi-davidson method for spectral low-rank preconditioning in computational electromagnetics problems, SEMA Journal 67 (1) (2015) 39–50. doi:10.1007/s40324-014-0025-6.

[17] B. Carpentieri, I. Duff, L. Giraud, G. Sylvand, An embedded iterative scheme in electromagnetism, in: R. Wyrzykowski, J. Dongarra, M. Paprzy-

cki, J. Wasniewski (Eds.), Parallel Processing and Applied Mathematics, Vol. 3019 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 977–984. doi:10.1007/978-3-540-24669-5_126.
URL http://dx.doi.org/10.1007/978-3-540-24669-5_126

[18] K. Schloegel, G. Karypis, V. Kumar, Sourcebook of parallel computing, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003, Ch. Graph Partitioning for High-performance Scientific Simulations, pp. 491–541.
URL http://dl.acm.org/citation.cfm?id=941480.941499

[19] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing Co., Boston, 1996.

[20] M. Benzi, M. Tůma, Orderings for factorized sparse approximate inverse preconditioners, SIAM J. on Scientific Computing 21 (5) (2000) 1851–1868.

[21] M. Benzi, J. Marín, M. Tůma, Parallel preconditioning with factorized sparse approximate inverses, in: Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing 1999 (San Antonio, TX), SIAM, Philadelphia, PA, 1999, p. 5.

[22] Y. Saad, M. H. Schulz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing 7 (1986) 856–869.

[23] G. Karypis, V. Kumar, METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices (version 3.0), Tech. rep., University of Minnesota, Department of Computer Science and Army HPC Research Center (October 1997).

[24] M. Benzi, D. B. Szyld, A. van Duin, Orderings for incomplete factorization preconditioning of nonsymmetric problems, SIAM J. on Scientific Computing 20 (5) (1999) 1652–1670.

[25] I. S. Duff, A. M. Erisman, J. K. Reid, Direct Methods for Sparse Matrices, Oxford University Press, London, 1986.

[26] A. George, J. W. H. Liu, Computer Solution of Large Sparse Positive Definite Systems, Prentice-Hall, Englewood Cliffs, NJ., 1981.

[27] M. Benzi, J. C. Haws, M. Tůma, Preconditioning highly indefinite and nonsymmetric matrices, SIAM J. on Scientific Computing 22 (4) (2000) 1333–1353.

[28] I. S. Duff, J. Koster, The design and use of algorithms for permuting large entries to the diagonal of sparse matrices, SIAM J. Matrix Anal. 20 (1999) 889–901.

p	*Symm*	*Tr*	ρ	*Tp*	*Tsol*	*iter*
1	no	-	5.5			†
	mmd	0.4	1.90	1.95	33.4	370
2	no	-	4.4	5.2	38.9	378
	mmd	0.5	1.88	2.2	16.8	187
	qmd	0.5	1.88	2.6	17.8	195
	rcm	0.5	2.04	2.23	17.2	188
	nd	0.5	2.6	2.7	32.3	348
3	no	-	3.80	5.2	37.4	376
	mmd	0.4	1.87	2.4	16.9	188
	qmd	0.4	1.88	2.6	16.3	181
	rcm	0.4	2.01	3.4	17.2	190
	nd	0.4	2.3	3.1	18.5	199
4	no	-	3.2	5.3	35.3	370
	mmd	0.4	1.90	2.7	17.1	193
	qmd	0.4	1.90	3.2	17.2	193
	rcm	0.3	2.13	2.8	18.5	204
	nd	0.4	2.2	2.9	30.4	338
5	no	-	3.0	2.7	18.8	199
	mmd	0.2	1.96	2.5	16.7	190
	qmd	0.3	2.1	2.8	17.4	193
	rcm	0.2	2.2	2.9	17.3	194
	nd	0.3	2.4	2.8	28.9	315

Table 2: Effect of symmetric reorderings for the matrix CN3, ILUT(0.02).

p	$Symm$	Tr	ρ	Tp	$Tsol$	$iter$
1	no	-	4.7	4.4	81.2	796
	mmd	0.4	3.2	1.9	53.8	570
2	no	-	2.7	3.7	8.1	112
	mmd	0.4	2.3	3.4	10.4	107
	qmd	0.4	2.5	3.2	9.1	115
	rcm	0.4	2.1	2.7	8.0	111
	nd	0.4	2.6	2.5	10.1	117
3	no	-	2.5	3.3	10.9	110
	mmd	0.4	2.1	3.3	8.2	102
	qmd	0.4	2.2	3.3	9.0	105
	rcm	0.4	2.0	3.4	9.7	104
	nd	0.4	2.4	3.8	9.9	116
4	no	-	2.5	2.2	10.5	110
	mmd	0.5	2.0	2.1	8.0	107
	qmd	0.5	2.3	3.3	10.9	110
	rcm	0.5	2.0	2.9	9.7	102
	nd	0.5	2.4	2.1	11.7	112
5	no	-	2.4	4.4	10.4	106
	mmd	0.4	2.1	3.2	7.8	103
	qmd	0.4	2.2	3.0	7.8	104
	rcm	0.4	2.2	2.3	7.5	103
	nd	0.4	2.3	3.4	11.2	112

Table 3: Effect of symmetric reorderings for the matrix CETAF3, ILUT(0.04).

Non-Symm	*Symm*	*Tr*	ρ	*Tp*	*Tsol*	*iter*
no	no	-	2.5	3.3	10.9	110
	mmd	0.4	2.1	3.3	8.2	102
	qmd	0.4	2.2	3.3	9.0	105
	rcm	0.4	2.0	3.4	9.7	104
mst	no	0.1	2.5	4.5	9.7	110
	mmd	0.3	2.1	3.1	9.7	104
	qmd	0.4	2.2	3.7	10.4	104
	rcm	0.4	2.1	3.6	11.4	103
mpt	no	0.1	2.5	4.2	9.7	110
	mmd	0.4	2.1	3.0	9.0	104
	qmd	0.5	2.2	3.5	9.9	104
	rcm	0.4	2.0	2.9	9.7	103

Table 4: Effect of nonsymmetric reorderings for the matrices CETAF3, p=3, ILUT(0.04).

Non-Symm	*Symm*	*Tr*	ρ	*Tp*	*Tsol*	*iter*
no	no	-	3.8	5.2	37.4	376
	mmd	0.3	1.87	2.9	19.7	194
	qmd	0.3	1.81	2.7	22.5	275
	rcm	0.4	1.89	3.6	18.4	182
mst	no	0.12	3.86	4.5	37.9	378
	mmd	0.4	1.86	2.7	16.8	188
	qmd	0.33	1.88	3.7	15.3	181
	rcm	0.27	1.89	2.3	18.3	192
mpt	no	0.12	3.83	5.4	38.0	380
	mmd	0.4	1.87	2.4	16.9	188
	qmd	0.4	1.88	2.6	16.3	181
	rcm	0.4	2.01	3.4	17.2	190

Table 5: Effect of nonsymmetric reorderings for the matrices CN3, p=3, ILUT(0.02).

Non-Symm	*Symm*	*Tr*	ρ	*Tp*	*Tsol*	*iter*
no	no	-	2.4	9.9	35.1	156
	mmd	0.7	2.0	8.7	25.7	118
	qmd	1.0	2.1	10.3	28.6	130
	rcm	0.7	2.1	10.8	31.8	143
mst	no	0.3	2.5	10.9	36.8	162
	mmd	1.0	2.1	9.8	26.6	122
	qmd	1.1	2.2	10.6	30.5	138
	rcm	1.1	2.1	7.5	31.4	142
mpt	no	0.3	2.5	11.5	36.7	162
	mmd	1.0	2.1	10.4	26.6	122
	qmd	1.1	2.2	8.1	30.5	138
	rcm	1.0	2.1	7.5	31.4	142

Table 6: Effect of nonsymmetric reorderings for the matrices CETAF5, p= 3, ILUT(0.05).

Non-Symm	*Symm*	*Tr*	ρ	*Tp*	*Tsol*	*iter*
no	no	-	3.16	12.2	129.2	555
	mmd	0.8	1.62	6.4	33.3	153
	qmd	1.3	1.85	7.7	33.4	125
	rcm	1.0	1.87	7.9	43.8	189
mst	no	0.3	3.47	9.0	301.4	1199
	mmd	1.1	1.67	5.1	29.3	135
	qmd	1.7	1.73	5.2	33.8	154
	rcm	1.1	1.85	5.3	41.3	185
mpt	no	0.3	3.40	10.1	199.8	798
	mmd	1.2	1.66	6.2	32.7	150
	qmd	1.7	1.72	7.3	37.4	152
	rcm	1.2	1.86	7.8	44.7	182

Table 7: Effect of nonsymmetric reorderings for the matrices CN5, p= 3, ILUT(0.04).

Matrix	p	τ	Tr	ρ	Tp	$Tsol$	$iter$
CN2	1	0.001	0.2	2.4	1.9	14.1	325
	2	0.001	0.2	1.8	1.6	10.9	253
	3	0.001	0.2	1.8	1.7	10.9	253
	4	0.001	0.2	1.8	2.0	10.9	245
	5	0.001	0.2	1.8	2.1	10.9	238
CN5	1	0.04	0.9	2.0	4.8	39.5	168
	2	0.04	0.8	1.8	6.4	31.8	157
	3	0.04	0.8	1.6	6.6	30.3	153
	4	0.04	0.8	1.7	7.1	28.5	133
	5	0.04	0.7	1.6	7.1	29.8	140
CETAF5	1	0.06	0.98	3.1	3.8	82.5	346
	2	0.06	0.8	2.2	7.2	31.3	140
	3	0.05	0.7	2.0	8.7	25.7	118
	4	0.05	0.8	2.1	8.3	26.3	121
	5	0.05	0.6	2.1	8.6	25.7	119
CETAF10	1	0.05	5.5	3.3	20.9	813	935
	2	0.05	5.1	2.6	20.4	475	589
	3	0.05	4.3	2.4	24.0	318	398
	4	0.05	3.7	2.5	30.5	362	455
	5	0.05	3.5	2.4	40.3	344	432
	6	0.05	3.4	2.4	43.9	364	461

Table 8: Effect of the number of partitions for the matrix CN2, CN5, CETAF5 and CETAF10. τ indicates the ILUT dropping threshold. Matrices reordered only with multiple minimum degree.

matriz	ρ	*Tp*	*Tsol*	*Total time*	*iter*
CETAF3	1.9	24.2	9.9	34.1	169
CETAF5	2.3	148.1	30.2	32.5	189
CETAF10	2.4	437.5	592.0	1029.5	689
CN2	1.8	12.8	8.0	20.8	197
CN3	1.9	23.6	14.9	38.5	186
CN5	1.7	94.1	84.0	178.1	399

Table 9: Results obtained with SPAI for the tested matrices.

Matrix	p	*Schur size*	*Ratio*	*iter*	*Total time*
CETAF3	2	241	5.9	107	14.2
	3	425	2.1	102	11.6
	4	641	1.0	107	10.6
	5	714	0.7	103	11.4
CETAF5	2	560	3.1	140	39.3
	3	776	1.8	118	35.1
	4	1090	0.9	121	35.4
	5	1278	0.6	119	34.9
CETAF10	2	719	6.5	589	500.5
	3	1153	2.6	398	343.5
	4	1736	1.2	455	396.2
	5	1850	0.9	432	387.8
	6	2326	0.6	461	411.3
CN2	2	95	10.2	253	12.7
	3	186	3.3	253	12.8
	4	246	1.8	245	13.0
	5	304	1.1	238	13.1
CN3	2	177	8.0	187	19.5
	3	260	3.5	188	19.7
	4	470	1.4	193	20.2
	5	592	0.8	190	19.9
CN5	2	176	13.7	157	39.0
	3	333	4.7	153	37.7
	4	700	1.5	133	35.4
	5	889	0.9	140	37.6

Table 10: Schur complement block sizes of the p-way partitionings, iterations and total CPU time in seconds for the matrices tested. *Ratio* indicates the average diagonal block size to Schur block size ratio. Matrices reordered only with multiple minimum degree.