ACADEMIC PRESS

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational Physics 185 (2003) 158–175

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

Incomplete LU preconditioning for large scale dense complex linear systems from electromagnetic wave scattering problems

Jeonghwa Lee [a,1], Jun Zhang [a,*,2], Cai-Cheng Lu [b,3]

[a] *Laboratory for High Performance Scientific Computing and Computer Simulation, Department of Computer Science, University of Kentucky, 773 Anderson Hall, Lexington, KY 40506–0046, USA*

[b] *Department of Electrical and Computer Engineering, University of Kentucky, Lexington, KY 40506-0046, USA*

Received 1 May 2002; received in revised form 20 October 2002; accepted 14 November 2002

Abstract

We consider the preconditioned iterative solution of large dense linear systems, where the coefficient matrix is a complex valued matrix arising from discretizing the integral equation of electromagnetic scattering. For some scattering structures this matrix can be poorly conditioned. The main purpose of this study is to evaluate the efficiency of a class of incomplete LU (ILU) factorization preconditioners for solving this type of matrices. We solve the electromagnetic wave equations using the BiCG method with an ILU preconditioner in the context of a multilevel fast multipole algorithm (MLFMA). The novelty of this work is that the ILU preconditioner is constructed using the near part block diagonal submatrices generated from the MLFMA. Experimental results show that the ILU preconditioner reduces the number of BiCG iterations substantially, compared to the block diagonal preconditioner. The preconditioned iteration scheme also maintains the computational complexity of the MLFMA, and consequently reduces the total CPU time.
© 2002 Elsevier Science B.V. All rights reserved.

Keywords: Krylov subspace methods; ILU preconditioning; Multilevel fast multipole algorithm; Electromagnetic scattering

[*] Corresponding author. Tel.: +1-859-257-3892; fax: +1-859-323-1971.
E-mail addresses: leejh@engr.uky.edu (J. Lee), jzhang@cs.uky.edu (J. Zhang), cclu@engr.uky.edu (C.-C. Lu).
URLs: http://www.csr.uky.edu/~leejh, http://www.cs.uky.edu/~jzhang, http://www.engr.uky.edu/~cclu.

[1] This author's research work was supported in part by the US National Science Foundation under Grant CCR-0092532.

[2] This author's research work was supported in part by the US National Science Foundation under Grants CCR-9902022, CCR-9988165, CCR-0092532, and ACI-0202934, by the US Department of Energy Office of Science under Grant DE-FG02-02ER45961, by the Japanese Research Organization for Information Science and Technology (RIST), and by the University of Kentucky Research Committee.

[3] This author's research work was supported in part by the US National Science Foundation under Grant ECS-0093692, and by the US Office of Naval Research under Grant N00014-00-1-0605.

0021-9991/02/$ - see front matter © 2002 Elsevier Science B.V. All rights reserved.
doi:10.1016/S0021-9991(02)00052-9

1. Introduction

The electromagnetic wave scattering by three-dimensional (3D) arbitrarily shaped dielectric and conducting objects can be obtained by finding the solution of an integral equation. The solution to electromagnetic wave interaction with material coated objects has applications in radar cross-section (RCS) prediction for coated targets, printed circuit, and microstrip antenna analysis [14,29,31]. This problem can be studied using integral equation solvers. For example, to calculate radar scattering by a conducting target coated by dielectric material, the hybrid surface and volume integral equations (VIEs) can be used [18]. In this case, a surface integral equation (SIE) is formed for the conducting surfaces and a VIE is formed for the dielectrics. The integral equations are discretized into a matrix equation by the method of moments (MoM) [18,21,30]. With this procedure, we obtain a linear system of the form

$$Ax = b, \tag{1}$$

where the coefficient matrix A is a large scale, dense, and complex valued matrix for electrically large targets.

The matrix equation (1) can be solved by numerical matrix equation solvers. Typical matrix solvers can be categorized into two classes. The first class is the direct solution methods, of which the Gauss elimination method is representative. The second class is the iterative solution methods, of which the Krylov subspace methods are considered to be the most effective ones currently available [1,24]. The biconjugate gradient (BiCG) method is one of the many Krylov subspace methods [15].

Direct solution methods are very expensive in both memory space and CPU time for large size problems. For the Gauss elimination method the floating point operation count is on the order of $\mathrm{O}(N^3)$, where N is the number of unknowns (columns) of the matrix A. In contrary, the complexity of the BiCG type iterative methods is on the order of $\mathrm{O}(N^{\mathrm{iter}}N^2)$ if the convergence is achieved in N^{iter} iterations. The Krylov subspace methods such as BiCG require the computation of some matrix–vector product operations at each iteration, which account for the major computational cost of this class of methods. The fast multipole method (FMM) speeds up the matrix–vector product operations when it is used to solve the matrix equation iteratively. The FMM approach reduces the computational complexity of a matrix–vector product from $\mathrm{O}(N^2)$ to $\mathrm{O}(N^{1.5})$ [11,12,22]. With the multilevel fast multipole algorithm (MLFMA) the computational complexity is further reduced to $\mathrm{O}(N \log N)$ [9,17,27,28].

It should be noted that a matrix problem involving N unknowns may be solved in $CN^{\mathrm{iter}}N^{Ax}$ floating point operations, where C is a constant depending on the implementation of a particular iterative method [3,11,12,24], and N^{Ax} is the floating point operations needed for each matrix–vector multiplication. For many realistic problems, N^{iter} depends on both the iterative solver and the target properties (shape and material). For example, a problem with an open-ended cavity needs much more number of iterations than that with a solid conducting box of the same size. Since N^{iter} is a proportional factor in the CPU counter, to further reduce the total CPU time, it is necessary to reduce the number of iterations of the iterative solvers. Hence preconditioning techniques, which may speed up the convergence rate of the Krylov subspace methods, are needed in this application. There are various preconditioning techniques in existence, most of them are developed in the context of iteratively solving linear equations involving large sparse matrices [19,24].

In the MLFMA implementation, the global coefficient matrix A is not explicitly available. Thus the strategy of extracting a sparsity pattern by dropping small magnitude entries from the dense coefficient matrix to form a sparse matrix as a basis for constructing a preconditioner is not feasible. We propose to use an incomplete lower–upper (ILU) triangular factorization with a dual dropping strategy (ILUT [23]) to construct a preconditioner from the near part matrix of the coefficient matrix in the MLFMA implementation. The near part matrix is naturally available in the MLFMA. By not using a static (prespecified) sparsity pattern, we hope to capture the most important (large magnitude) entries in the incomplete LU factorization, while not to consume a large amount of memory. This is achieved by dynamically dropping small magnitude entries during the computation and by only allowing a fixed number of nonzeros kept in each row of the L and U factors.

In our experimental study, we use the BiCG method as the iterative solver, coupled with three preconditioning strategies to solve a few study cases of representative electromagnetic scattering problems. We compare mainly the preconditioning effect of different strategies. The preconditioning strategies that we consider are (a) no preconditioner (equivalent to using the identity matrix as the preconditioner), (b) the block diagonal preconditioner, and (c) the incomplete LU triangular factorization preconditioner. We show by numerical simulations that the BiCG method with the ILU preconditioner reduces the number of iterations significantly, compared to the cases with the block diagonal preconditioner as well as without a preconditioner.

This paper is organized as follows. Section 2 gives a concise introduction to the hybrid integral equation approach in electromagnetic scattering and the MLFMA. In Section 3, we outline the preconditioned BiCG method and discuss in detail the ILUT preconditioner. Extensive numerical experiments with a few electromagnetic scattering problems are conducted in Section 4, and in Section 5, we explain and analyze the numerical data obtained in our experiments. Some concluding remarks are given in Section 6.

2. Discretization of integral equation and fast multipole method

The hybrid integral equation approach combines the VIE and the surface integral equation (SIE) to model the scattering and radiation by mixed dielectric and conducting structures [18,26]. For example, when a radome is applied to an antenna, the combined system consists of both dielectrics and conductors. Hence, the hybrid surface–volume integral equation is ideal for this problem [9]. The VIE is applied to the material region (V) and the SIE is enforced over the conducting surface (S). The integral equations can be formally written as follows:

$$\{L_S(r,r')\cdot J_S(r') + L_V(r,r')\cdot J_V(r')\}_{\tan} = -E^{\rm inc}_{\tan}(r), \quad r\in S,$$

$$-E + L_S(r,r')\cdot J_S(r') + L_V(r,r')\cdot J_V(r') = -E^{\rm inc}(r), \quad r\in V,$$

where $E^{\rm inc}$ stands for the excitation field produced by an instant radar, the subscript "tan" stands for taking the tangent component from the vector it applies to, and $L_\Omega, (\Omega = S, V)$, is an integral operator that maps the source J_Ω to electric field $E(r)$ and it is defined as

$$L_\Omega(r,r')\cdot J_\Omega(r') = \mathrm{i}\omega\mu_b \int_{\Omega'} \left(I + k_b^{-2}\nabla\nabla\right) G(r,r')\cdot J_\Omega(r')\,\mathrm{d}\Omega'.$$

Here $G(r,r') = \mathrm{e}^{\mathrm{i}k_b|r-r'|}/(4\pi|r-r'|)$ is the 3D scalar Green's function for the background media, and $\mathrm{i} = \sqrt{-1}$. It should be pointed out that E is related to J_V in the above integral equations by $J_V = \mathrm{i}\omega(\epsilon_b - \epsilon)E$. This results in a very general model as all the volume and surface regions are modeled properly. The advantage of this approach is that in the coated object scattering problems, the coating material can be inhomogeneous, and in the printed circuit and microstrip antenna simulation problems the substrate can be of finite size. The simplicity of the Green's function in both the VIE and the SIE has an important impact on the implementation of the fast solvers. However, the additional cost here is the increase in the number of unknowns since the volume that is occupied by the dielectric material is meshed. This results in larger memory requirement and longer solution time in solving the corresponding matrix equation. But this deficiency can be overcome by applying fast integral equation solvers such as the MLFMA [9].

We follow the general steps of the MoM to discretize the hybrid surface–volume integral equations. This involves the following steps: (a) Use a collection of small patches to represent the surface S, and a collection of small cells to represent the volume V. (b) Select a set of surface basis functions f_n^S to expand the surface current, and select a set of volume basis functions to expand the volume function $\mathrm{i}\omega\epsilon E$. Since the volume basis function we use is continuous across the cells, we assume $i\omega\epsilon E$ to be the unknown function, from which E and J_V can be determined. (c) Substitute the approximate representation of the unknowns into the integral equation, and test the resultant integral equation with a set of testing functions. As a result, the hybrid integral equations are converted into a matrix equation that is formally written as

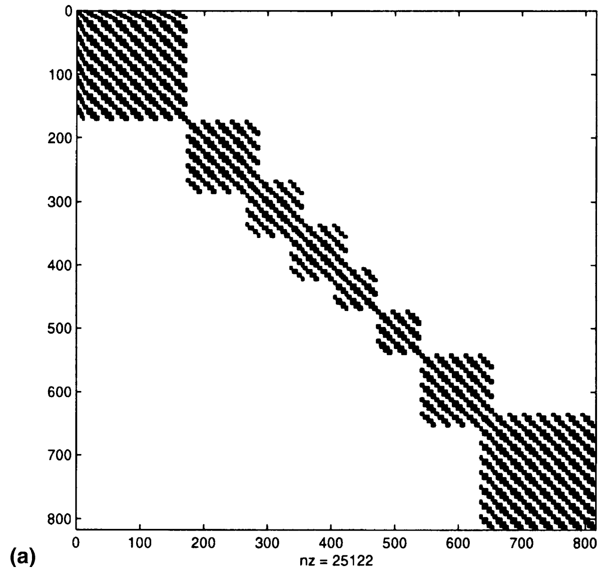
$$\begin{bmatrix} Z^{SS} & Z^{SV} \\ Z^{VS} & Z^{VV} \end{bmatrix} \cdot \begin{bmatrix} a^S \\ a^V \end{bmatrix} = \begin{bmatrix} U^S \\ U^V \end{bmatrix}, \tag{2}$$

where a^S and a^V stand for the vectors of the expansion coefficients for the surface current and the volume function, respectively [9,18], and the matrix elements can be generally written as

$$Z_{ji} = \mathrm{i}\omega\mu_b \int_{\Omega} \mathrm{d}\Omega f_j^{\Omega}(r) \cdot \int_{\Omega'} \mathrm{d}\Omega' (I + k_b^{-2}\nabla\nabla) G(r,r') \cdot \chi(r') f_i^{\Omega'}.$$

The material function $\chi(r') = 0$ if Ω' is a surface patch, and $\chi = (\epsilon/\epsilon_b - 1)$ if Ω' is a volume cell. It can be seen that the coefficient matrix arising from discretized hybrid integral equations is nonsymmetric. Once the matrix equation (2) is solved by numerical matrix equation solvers, the expansion coefficients a^S and a^V can be used to calculate the scattered field and RCS. In antenna analysis problems the coefficients can be used to retrieve the antenna's input impedance and calculate the antenna's radiation pattern. In the following, we use A to denote the coefficient matrix in Eq. (2), $x = [a^S, a^V]^{\mathrm{T}}$, and $b = [U^S, U^V]^{\mathrm{T}}$ for simplicity.

The basis function is an elementary current that has local support. To solve the matrix equation by an iterative method, the matrix–vector multiplications are needed at each iteration. Physically, a matrix–vector multiplication corresponds to one cycle of interactions between the basis functions. The basic idea of the FMM is to convert the interaction of element-to-element to the interaction of group-to-group. Here a



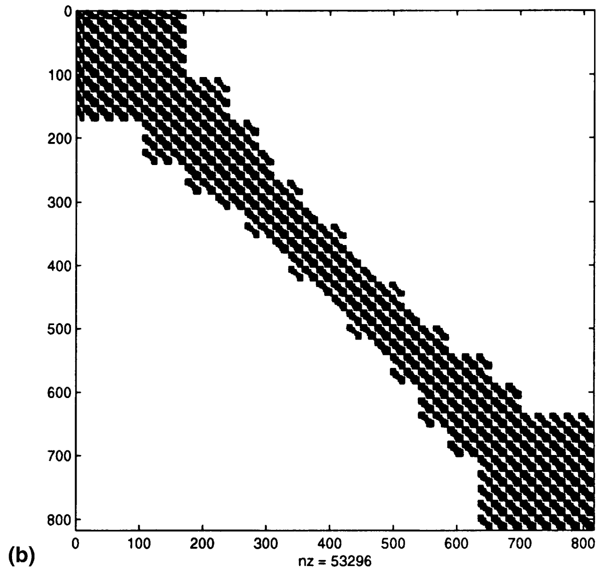


Fig. 1. The sparse data structure of a dense matrix A from electromagnetic scattering. (a) The block diagonal part. (b) The block diagonal and near-diagonal part.

group includes the elements residing in a spatial box. The mathematical foundation of the FMM is the addition theorem for the free-space scalar Green's function [11]. Using the addition theorem, the matrix–vector product Ax can be written as (see Appendix A)

$$Ax = (A_D + A_N)x + V_f \Lambda V_s x, \tag{3}$$

where V_f, Λ, and V_s are sparse matrices. In fact, the dense matrix A can be structurally divided into three parts, A_D, A_N, and $A_F = V_f \Lambda V_s$. A_D is the block diagonal part of A, A_N is the block near-diagonal part of A, and A_F is the far part of A. Here the terms "near" and "far" refer to the distance between two groups of elements. Fig. 1 shows the sparse data structure of a partitioned dense matrix A from one of our examples, the P1A case in Table 1. Fig. 1(a) shows A_D, Fig. 1(b) shows $A_D + A_N$, and the far part of A is scattered in the rest of the area of Fig. 1(b).

To implement the FMM, the spatial region of all the basis functions are subdivided into small cubic boxes. Then the scattered field of different scattering centers within a group are translated into a single center called group center. This is the aggregation process. Hence, the number of scattering centers is reduced. If two groups are separated far enough, the interactions between them are carried out through translation. After the interactions among the groups are done, the field received by a group is redistributed to the group

Table 1
Information about the matrices used in the experiments (all length units are in λ_0, the wavelength in free-space)

Cases	Level	Unknowns	Matrices	Nonzeros	Target size and description
P1A	4	816	A	665,856	6×2
			A_D	25,122	Conducting plate
			$A_D + A_N$	53,296	
P1B	4	1416	A	2,005,056	$2.98824 \times 2 \times 0.1$
			A_D	66,384	Dielectric plate over conducting plate
			$A_D + A_N$	155,616	
S1A	4	4800	A	23,040,000	$0.6 \times 0.2 \times 0.1$
			A_D	514,196	Conducting sphere (partial)
			$A_D + A_N$	1,795,306	
S1B	4	7576	A	57,395,776	$0.62 \times 0.0747217 \times 0.12$
			A_D	1,506,932	Conducting sphere with dielectric coating (partial)
			$A_D + A_N$	5,125,932	
P2A	3	625	A	390,625	$0.266 \times 0.116 \times 0.001587$
			A_D	64,753	Microstrip antenna (no substrate)
			$A_D + A_N$	117,121	
P2B	3	1683	A	2,832,489	$0.266 \times 0.116 \times 0.001587$
			A_D	441,479	Microstrip antenna
			$A_D + A_N$	773,181	
S2A	4	10,800	A	116,640,000	$5 \times 5 \times 5$
			A_D	555,200	Large conducting sphere
			$A_D + A_N$	1,590,200	
S2B	4	32,400	A	1,049,760,000	$5.2 \times 5.2 \times 5.2$
			A_D	5,382,304	Large sphere with dielectric coating
			$A_D + A_N$	14,756,720	
P3A	7	100,800	A	10,160,640,000	22.25×22.25
			A_D	3,571,808	Antenna array
			$A_D + A_N$	7,211,632	

members. This is the disaggregation process [27]. In the FMM, the matrix elements are calculated differently depending on the distance between a testing function and a basis function. For near-neighbor matrix elements (those in A_D and A_N), the calculation remains the same as in the MoM procedure. However, those elements in A_F are not explicitly computed and stored. Hence they are not numerically available in the FMM. It can be shown that with optimum grouping, the number of nonzero elements in the sparse matrices in Eq. (3) are all on the order of $N^{1.5}$, and hence the operation count to perform Ax is $O(N^{1.5})$.

If the above process is implemented in multilevel, the total cost (in memory and CPU time) can be reduced further to be proportional to $N \log N$ for one matrix–vector multiplication. Due to this reduced computational complexity, the memory and CPU time savings can be orders of magnitude compared with the direct dense matrix multiplication methods of $O(N^2)$ if N is very large [17,20].

3. Iterative methods and preconditioners

There have been a few preconditioning techniques recently developed for solving dense linear systems arising from discretized integral equations in electromagnetic applications and in other applications [5,7]. Until recently, preconditioning techniques for solving dense matrices are less extensively studied [7], compared to those for the sparse matrices. In the electromagnetic scattering simulation field, the diagonal and block diagonal preconditioners are considered by a few authors [17,27,28] in the context of the MLFMA. The individual blocks (of the block diagonal matrix) are numerically available and are easy to invert. More recently developed strategies are related to the sparsity pattern based ILU factorization preconditioning [25] and to sparse approximate inverse preconditioning [6]. Some of them are based on the pattern of a sparsified coefficient matrix [2,4,6]. Others are constructed using the geometric or topological information of the underlying problems [20,25]. Most of these preconditioning techniques, such as the ILU(0), rely on a fixed sparsity pattern, obtained from the sparsified coefficient matrix by dropping small magnitude entries. The ILU(0) preconditioner, which uses a static sparsity pattern and is constructed from a sparsified matrix, is shown to be ineffective in some electromagnetic scattering problems [5]. On the other hand, some sparse approximate inverse techniques need access to the full coefficient matrix (to construct a sparsified matrix), which is not available in the FMM. In many cases, the sparse approximate inverse techniques are far more expensive to construct than the ILU type preconditioning techniques. This is the case when the inherent parallelism in constructing Frobenius-norm based sparse approximate inverse preconditioners is not exploited.

We solve the dense linear system (1) by using the preconditioned BiCG method. We will evaluate two preconditioners, the ILU preconditioner with a dual dropping strategy (a fill-in parameter and a drop tolerance) [23] and the block diagonal preconditioner, using the trivial identity matrix preconditioner as comparison. The block diagonal preconditioner for the MLFMA is implemented in [28]. Due to space limit, we will not be able to evaluate other type of preconditioning techniques. The efficiency and examples of the ILU(0) and some sparse approximate inverse preconditioners in the FMM implementation can be found in [20,25].

By the block diagonal preconditioning, we construct a preconditioner A_D^{-1} from the block diagonal matrix A_D, and then apply the preconditioner A_D^{-1} to the linear system (1). That is, $A_D^{-1}Ax = A_D^{-1}b$. Since A_D is a block diagonal matrix, each individual block can be inverted independently.

By the ILU preconditioning, we construct a preconditioner M^{-1} from the ILU factorization of the near part matrix $(A_D + A_N)$, and then apply the preconditioner M^{-1} to the linear system (1). That is, $M^{-1}Ax = M^{-1}b$.

It is well known that a simple preconditioner, such as the ILU(0), which uses the sparsity pattern of A, may not be robust for solving some large scale ill-conditioned problems. Different high accuracy preconditioners have been proposed for solving sparse matrices [8,23,24,32]. Many of them are constructed using either an enlarged sparsity pattern or using some threshold value based drop tolerance to allow more fill-in in the construction phase. Some high accuracy ILU preconditioning strategies may not be able to control the amount of memory space needed a priori. Since memory cost is of vital importance in large scale electromagnetic scattering simulations, preconditioners using a lot of memory space are less useful. Therefore, we resort to a particular ILU factorization strategy which uses a dual dropping strategy (ILUT) to control both the computational cost and the memory cost. By using the ILUT, we anticipate to use not very much more than the amount of the memory space used to store the near part matrix $(A_D + A_N)$. Thus the memory and floating point operation complexity of the MLFMA is maintained.

The following algorithm of the ILUT factorization on a matrix $A = (a_{ij})_{N\times N}$ is due to Saad and is extracted from [23,24].

Algorithm 3.1. The ILUT Algorithm

```
0. Set $u_{1k} = a_{1k}, k = 1,\ldots,N$
1. For $i = 2,\ldots,N$, Do:
2.   $w = a_{i*}$
3.   For $k = 1,\ldots,i-1$ and when $w_k \neq 0$, Do:
4.     $w_k = w_k/u_{kk}$
5.     Apply a dropping rule to $w_k$
6.     If $w_k \neq 0$ then
7.       $w = w - w_k * u_{k*}$
8.     EndIf
9.   EndDo
10.  Apply a dropping rule to row $w$
11.  $l_{i,j} = w_j$ for $j = 1,\ldots,i-1$
12.  $u_{i,j} = w_j$ for $j = i,\ldots,N$
13.  $w = 0$
14. EndDo
```

In Algorithm 3.1, w_k is a work array and a_{i*} denotes the ith (current) row of the matrix A. The dual dropping strategy of the ILUT is implemented using the two parameters τ and p. The small entries of w_k (with respect to τ and relative to a certain norm of the current row) are dropped in line 5. In line 10, small entries are dropped again. A search algorithm is used to find out the largest p entries in magnitude. These p largest entries are kept, the others are dropped again. The total storage is bounded by $2pN$. Here τ controls the computational cost, and p controls the memory cost. By judiciously choosing the two parameters τ and p, we may be able to construct an ILU preconditioner that is effective and does not use much memory space.

When ILU factorization preconditioners are used in combination with an iterative process such as a Krylov subspace method, they tend to dictate the convergence behavior more than the iterative process itself [23,33]. In general, high accuracy ILU type preconditioners with a large amount of fill-in are more robust but cost more in construction than their lower accuracy counterparts do [33,34]. Since an ILUT preconditioner may be reused several times by solving the same matrices with several different right hand sides in electromagnetic scattering simulations, its construction cost can be amortized. However, it is important to note that the memory space is usually a bottleneck in large scale electromagnetic scattering simulations.

For completeness, we give the following algorithm of the preconditioned BiCG method with a preconditioner M^{-1}.

Algorithm 3.2. The Preconditioned BiCG Method

```
1. Compute $r_0 = b - Ax_0$. Choose $r_0^*$ such that $(r_0, r_0^*) \neq 0$.
2. Compute $z_0 = M^{-1}r_0$, $z_0^* = (M^{-1})^H r_0^*$.
```

```
3. Set $p_0 = z_0, p_0^* = z_0^*$
4. For $j = 0, 1, \dots,$ until convergence, Do:
5.    $\alpha_j = (r_j, z_j^*)/(Ap_j, p_j^*)$
6.    $x_{j+1} = x_j + \alpha_j p_j$
7.    $r_{j+1} = r_j - \alpha_j A p_j$
8.    $r_{j+1}^* = r_j^* - \bar{\alpha}_j A^H p_j^*$
9.    $z_{j+1} = M^{-1} r_{j+1}$
10.   $z_{j+1}^* = (M^{-1})^H r_{j+1}^*$
11.   $\beta_j = (r_{j+1}, z_{j+1}^*)/(r_j, z_j^*)$
12.   $p_{j+1} = r_{j+1} + \underline{\beta}_j p_j$
13.   $p_{j+1}^* = z_{j+1}^* + \bar{\beta}_j p_j^*$
14. EndDo
```

In Algorithm 3.2, the initial guess is x_0. Implicitly, the algorithm solves not only the original system (1) but also a dual linear system $A^H x^* = b^*$ with A^H (the complex conjugate transpose of A) if an update for x_j^* is included [24]. At each iteration, BiCG needs two matrix–vector products, one with A and one with A^H.

4. Numerical experiments

In this section, we present a number of numerical examples to demonstrate the efficiency of the ILUT preconditioner [23]. All cases are tested on one processor of an HP Superdome cluster at the University of Kentucky. The processor has 2 GB local memory and runs at 750 MHz. The code is written in Fortran 77 and is run in single precision.

To demonstrate the performance of our preconditioned BiCG solver, we calculate the RCS of different conducting geometries with and without coating. The geometries considered include plates, spheres, antennas and antenna arrays. The mesh sizes for all the test structures are about one tenth of a wavelength. The information describing the structures and the generated matrices is given in Table 1. The second column entitled "level" indicates the number of levels in the MLFMA. In most cases, three or four levels are used. For the largest case P3A, we have used seven levels. In Table 1 we also give the number of nonzeros of the full matrix A, the block diagonal matrix A_D, and the near part matrix $(A_D + A_N)$. Due to the space limit, we only report a few numerical results in this section. More detailed computational results can be found in a technical report [16].

There are two stopping criteria implemented in the code. One is the error-bound (residual norm), and the other is the limit of the number of iterations. The error-bound is to reduce the 2-norm residual by 10^{-3}, and the limit of the maximum number of iterations is set as 2000.

Here are the explanations of the notations shown in the data tables.

- *prec*: the preconditioner used with the BiCG method:
  - NONE: no preconditioner;
  - BLOCK: the block diagonal preconditioner [17,27,28];
  - ILUT: the incomplete LU preconditioner with a dual dropping strategy [23].
- τ: the threshold drop tolerance used in ILUT.
- p: the fill-in parameter used in ILUT.
- $ratio_f$: the sparsity ratio of the number of nonzeros of the L and U factors with respect to the number of nonzeros of the full matrix A.
- $ratio_n$: the sparsity ratio of the number of nonzeros of the L and U factors with respect to the number of nonzeros of the near part matrix $(A_D + A_N)$.
- LU_{cpu}: the CPU time in seconds for performing the incomplete LU factorization.

- it_{num}: the number of the (preconditioned) BiCG iterations.
- it_{cpu}: the CPU time in seconds for the iteration phase.
- *sol*: iterative solution quality, compared with the exact solution from the Gauss elimination (LUD) solver, or with the solution from the converged cases when the Gauss elimination solver cannot be used due to the memory limit:
  - same: converged within the given stopping criteria;
  - diff: did not converge within the given stopping criteria.

  We summarize our experimental results as follows:
  1. For the class of the problems we tested, the block diagonal preconditioner improves the BiCG convergence in only 3 (P1B, P2A, and P2B) out of the 9 cases. In the remaining 6 other cases, the block diagonal preconditioner actually hampers the BiCG convergence. It is our conclusion that the block diagonal preconditioner is not robust for solving this class of dense matrices arising from the combined hybrid integral formulation of the electromagnetic scattering problems.
  2. For almost all problems, the ILUT preconditioner improves the BiCG convergence significantly and reduces the total CPU time substantially. Our tests show that the ILUT preconditioner is robust for solving this class of dense matrices. We also see that different choices of the ILUT parameters τ and p may affect the convergence rate of the preconditioned iterative solver.

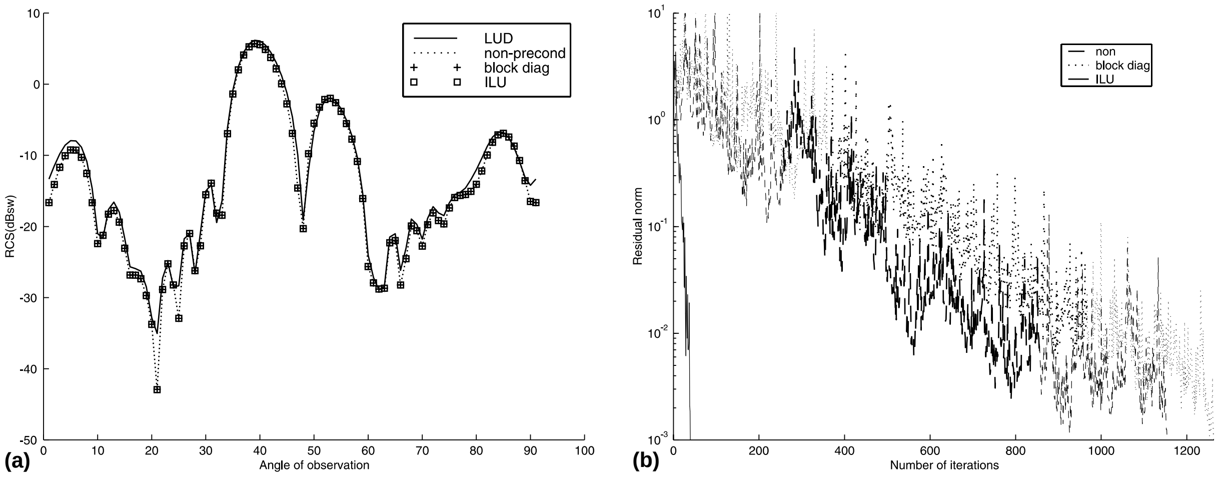


Fig. 2. The P1A case: ILUT ($\tau = 10^{-6}$, $p = 30$). (a) Solution comparison. (b) Convergence history.

Table 2
Comparison of solving the P1A case using different preconditioners

prec	τ	*p*	LU_{cpu}	$ratio_f$	$ratio_n$	it_{num}	it_{cpu}	*sol*
NONE						1153	97.75	same
BLOCK				0.03773		1265	107.02	same
ILUT	10^{-6}	30	0.06	0.07429	0.92817	39	3.51	same
		25	0.06	0.06263	0.78252	64	5.55	same
	10^{-5}	30	0.07	0.07427	0.92789	39	3.48	same
		25	0.05	0.06263	0.78248	59	5.16	same
	10^{-4}	30	0.05	0.07423	0.92735	40	3.60	same
		25	0.06	0.06259	0.78201	71	6.22	same
	10^{-3}	30	0.06	0.07384	0.92253	40	3.65	same
		25	0.06	0.06229	0.77826	66	5.86	same

3. It is worth pointing out that the memory cost of the ILUT preconditioner is reasonable. In most cases, the memory cost of the ILUT preconditioner is less than the cost of storing the near part matrix $(A_D + A_N)$, see the data in the column entitled $ratio_n$ in each table.
4. For most problems, the computation (construction) cost (LU_{cpu}) of the ILUT preconditioner is low, compared to the preconditioned iterative solution cost (it_{cpu}). The exceptions are found in the S1B and S2B cases. These two cases are related to modeling conducting spheres with dielectric coating.
5. For the converged cases, the approximate solutions computed from the preconditioned BiCG solver are comparable to those computed from the direct solver.
6. The accuracy of the computed solutions is affected by the level of the MLFMA.

Table 3
Numerical data for comparing different preconditioners ($\tau = 10^{-3}$ in ILUT)

Cases	*prec*	*p*	LU_{cpu}	$ratio_f$	$ratio_n$	it_{num}	it_{cpu}	*sol*
P1B	NONE					2000	223.20	same
	BLOCK			0.03311		1526	170.80	same
	ILUT	30	0.32	0.04249	0.54748	40	4.88	same
		25	0.27	0.03564	0.45921	72	8.62	same
		20	0.26	0.02868	0.36958	176	20.79	same
S1A	NONE					984	488.21	same
	BLOCK			0.02232		2000	1022.25	diff
	ILUT	300	16.15	0.12130	1.55673	102	84.52	same
		250	12.60	0.10184	1.30698	187	146.45	same
S1B	NONE					2000	1628.88	same
	BLOCK			0.02626		2000	1685.78	diff
	ILUT	500	198.16	0.12426	1.39138	69	117.64	same
		450	176.90	0.11354	1.27131	84	139.37	same
P2A	NONE					1057	9.19	same
	BLOCK			0.16577		107	0.97	same
	ILUT	60	0.12	0.18056	0.60220	18	0.39	same
		55	0.12	0.16749	0.55860	23	0.44	same
		50	0.11	0.15410	0.51395	29	0.54	same
P2B	NONE					2000	110.13	diff
	BLOCK			0.15586		199	12.86	same
	ILUT	100	2.08	0.11447	0.41934	38	3.88	same
		90	1.67	0.10352	0.37922	37	3.82	same
		80	1.58	0.09249	0.33885	65	6.09	same
S2A	NONE					317	873.65	same
	BLOCK			0.00476		2000	5530.08	diff
	ILUT	200	58.07	0.03683	2.70144	148	489.32	same
		150	38.68	0.02776	2.03594	224	702.20	same
S2B	NONE					690	3451.20	same
	BLOCK			0.00513		2000	10281.66	diff
	ILUT	300	3274.14	0.01845	1.31277	237	1756.59	same
		200	2236.59	0.01234	0.87819	286	1909.51	same
P3A	NONE					423	3018.69	same
	BLOCK			0.00227		625	4573.53	same
	ILUT	30	114.10	0.00355	0.77342	41	324.07	same
		20	112.70	0.00251	0.54629	45	347.90	same
		10	111.40	0.00137	0.29736	277	2060.10	same

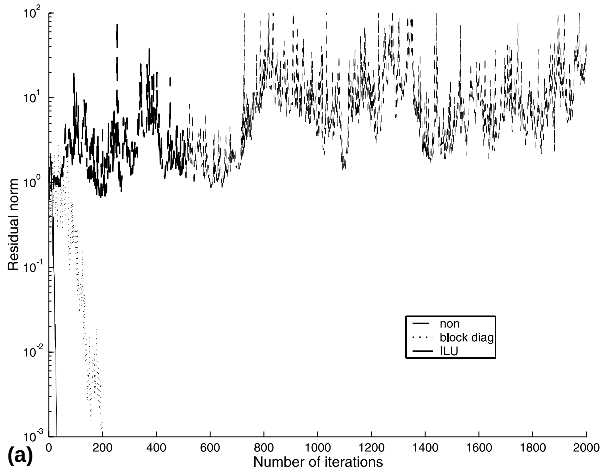
The solution graphs indicate the scattering pattern of the objects. If the number of unknowns are over 10,000 then we did not try to get the exact solution from the LUD decomposition because it is too time consuming.

Fig. 2(a) shows that the solutions of the P1A case computed with all three preconditioning strategies are similar and approximate to the exact solution. Fig. 2(b) demonstrates that the BiCG method with the block diagonal preconditioner converges more slowly than that without a preconditioner. This phenonmenon will be explained in Section 5. But the ILUT preconditioner is the most efficient one. More detailed comparisons can be found in Table 2, in which the ILUT with different τ and p parameters are examined, and compared with the block diagonal preconditioner, and with the case without a preconditioner. The ILUT with a range of parameter choices performs quite well in this test.

Test data in Table 3 indicate that the high accuracy ILUT preconditioner with more fill-in is shown to be more robust and more efficient. We also see that the sparsity ratio $ratio_n$ of the ILUT in most cases is less than 1 or very close to 1. The ILUT preconditioner with a suitable pair of τ and p parameters does not need a large amount of memory space. Thus, the memory and floating-point operation complexity of the MLFMA in each iteration is maintained. Since the number of iterations of the BiCG is decreased significantly using the ILUT, the total simulation time is reduced substantially.

Almost in all cases, the convergence behavior graphs (see Figs. 2(b) and 3) show that the BiCG method with the ILUT preconditioner converges very fast with a small number of iterations, compared to that with the block diagonal preconditioner and without a preconditioner.

In Table 4, we test the ILUT with a few parameters to work with the MLFMA with different levels for solving the S1A case. As the level of the MLFMA decreases, we find that the number of nonzeros in the near part matrix increases significantly. In fact, it is necessary to allow more fill-in entries in the ILUT



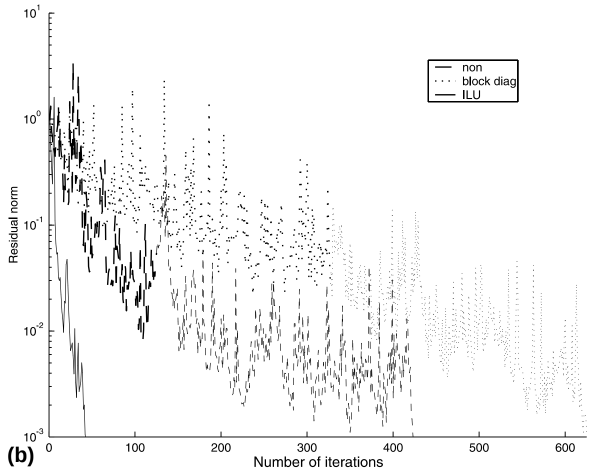
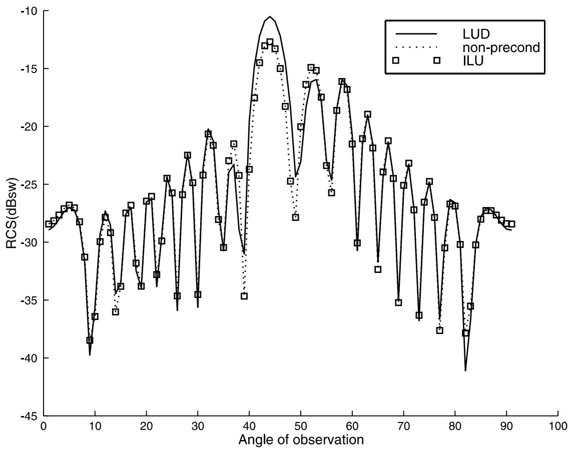


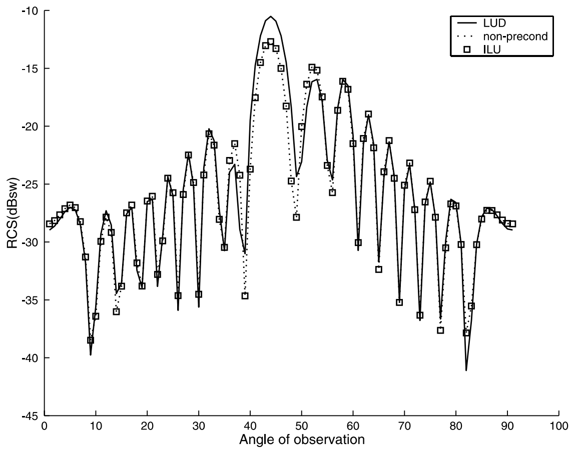
Fig. 3. Convergence history comparison in two test cases. (a) The P2B case, ILUT ($\tau = 10^{-6}, p = 100$). (b) The P3A case, ILUT ($\tau = 10^{-6}, p = 30$).

Table 4
Test results of the ILUT with different level of the MLFMA for solving the S1A case

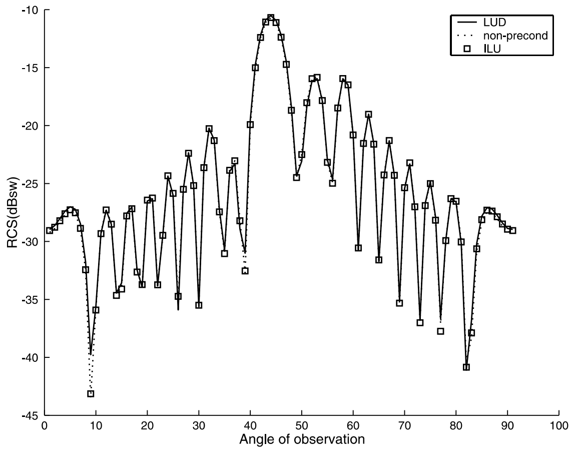
τ	p	Level	Nonzeros	$ratio_f$	$ratio_n$	it_{num}	it_{cpu}
10^{-6}	300	4	1,795,306	0.12131	1.55685	103	87.15
10^{-6}	500	3	6,366,904	0.19787	0.71605	116	153.27
10^{-6}	500	2	14,114,642	0.19788	0.32300	243	496.83



(a)



(b)



(c)

Fig. 4. The solution accuracy based on the MLFMA level and the error-bound (S1A): (a) level 4, error-bound 10^{-3}; (b) level 4, error-bound 10^{-6}; (c) level 2, error-bound 10^{-3}.

factorization to maintain good convergence, as the level of the MLFMA reduces. This is because the accuracy of the ILUT depends upon the relative amount of fill-in entries kept.

Fig. 4 shows the relations between the level of the MLFMA and the accuracy of the computed solutions. The ILUT preconditioned solutions are getting closer to the exact solutions as the number of the MLFMA level decreases. It is well known that the MLFMA is an approximation method [11]. As the number of level increases, the accuracy of the MLFMA approximation decreases. Fig. 4 demonstrates that the error-bound of the preconditioned iterative method used to solve the linear system does not have much effect on the accuracy of the computed solution.

5. Computational analysis

The convergence behavior of the Krylov subspace methods depends on the distribution of the eigenvalues and on the condition number of the coefficient matrix. Eigenvalues tightly clustered around a single point (away from the origin) provide fast convergence. While widely spread eigenvalues, especially around the origin, cause the convergence to be very slow. This is because a low degree polynomial with the value 1 at the origin cannot be small at a large number of such points [13,24].

Our experimental data in the previous section can be explained based on these theoretical facts. In Table 5, we report the largest and the smallest eigenvalues in magnitude of the original and the preconditioned matrices, and the condition number of the eigenvector matrices in the P1A case. According to [13,24], we can only link the convergence rate of GMRES (not BiCG) with the condition number of the eigenvector matrix X. For a given (diagonalizable) matrix A, it can be decomposed as $A = X\Sigma X^{-1}$, where $\Sigma = \mathrm{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ is the diagonal matrix of the eigenvalues and X is the matrix of the eigenvectors of the matrix A. The condition number of the eigenvector matrix X measures the normality of the matrix A. Although existing theoretical result is not for BiCG, it can be used to explain the convergence of the Krylov subspace methods (including BiCG) heuristically.

The condition number of the eigenvector matrix of the original dense matrix A is relatively large. The smallest magnitude eigenvalues of the original matrix (see Table 5 and Fig. 5(a)) are near the origin and they are almost spread on the negative side. The convergence of the BiCG method on the original matrix is slow.

The block diagonal preconditioned matrix ($(A_D)^{-1}A$) makes the spectrum spread around the origin (see Fig. 5(b)). This makes the preconditioned matrix more indefinite. These features cause a slow convergence of the BiCG method on the block diagonal preconditioned matrix. However, the eigenvalue decomposition shows that the condition number of the eigenvector matrix is smaller than that of the original matrix.

The ILUT preconditioned matrix ($(LU)^{-1}A$) is not like the block diagonal preconditioned matrix. Eigenvalues of the ILUT preconditioned matrix are shifted to the right-hand side of the origin and they are away from zero. They are also clustered around 1. This is clearly shown in Fig. 5(c). The smallest eigenvalue in magnitude has the real part of 0.4816, see Table 5. In addition, the condition number of the eigenvector matrix X is significantly decreased (the ILUT preconditioned matrix is close to normal). These explain the good convergence behavior of the BiCG method on this ILUT preconditioned matrix.

Table 5
Extreme eigenvalues and conditioning information of the matrices in the P1A case

Matrices	λ_{largest}	$\lambda_{\text{smallest}}$	$cond(X)$
A	$-0.0768 - 18.0300i$	$0.0527i$	531.2743
$(A_D)^{-1}A$	$6.7660 + 0.1193i$	$-0.1861 - 0.0020i$	37.2237
$(LU)^{-1}A$	$4.3659 + 0.2492i$	$0.4816 + 0.0126i$	9.9971

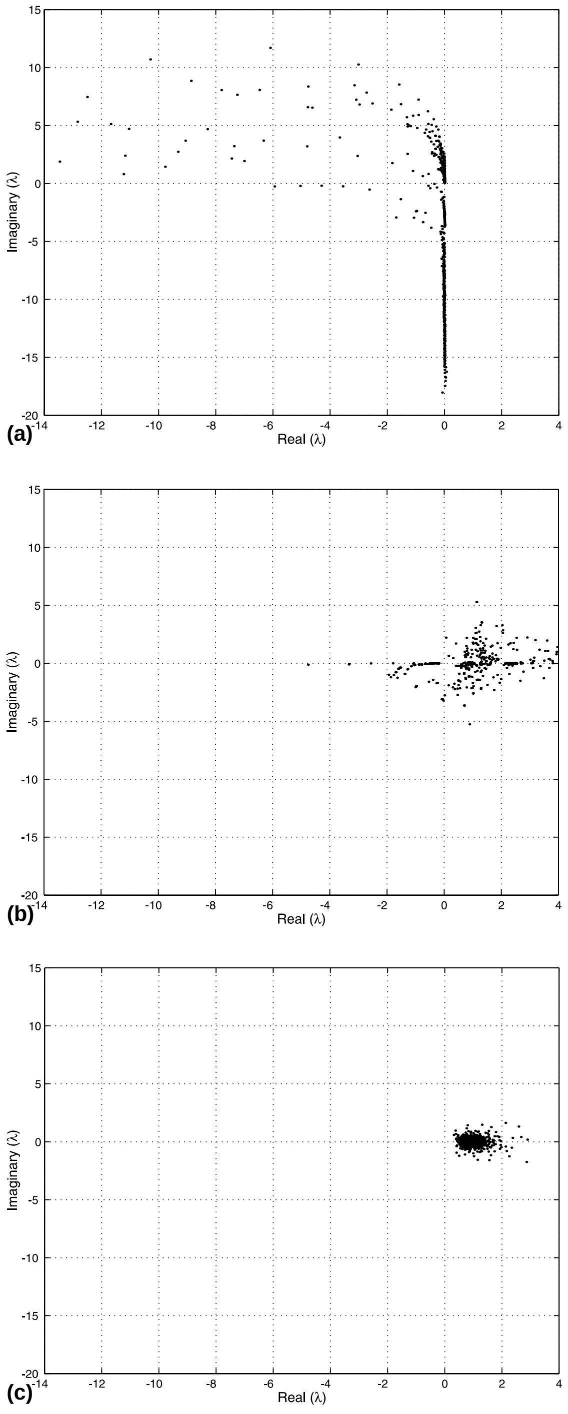


Fig. 5. Clustering of the eigenvalues in the P1A case: (a) eigenvalues of A; (b) eigenvalues of $(A_D)^{-1}A$; (c) eigenvalues of $(LU)^{-1}A$.

Accuracy and stability of the ILUT factorization. If the sparsity ratio $ratio_n$ of the ILUT preconditioner is very small the preconditioned BiCG iteration may not converge within the given stopping criteria or diverge. There are two possible reasons for the ILU type preconditioners to perform poorly for solving certain type of matrices. Of course, if the ILU preconditioner is not accurate, such as using large τ values and small p values in the ILUT preconditioner, the preconditioning matrix M is a poor approximation to the near part matrix $(A_D + A_N)$ (or to the coefficient matrix A). It is expected that the preconditioning effect of these poor quality ILU preconditioners is not good. Such poor quality ILU preconditioners due to insufficient amount of fill-in can usually be improved by allowing more fill-in, i.e., by choosing a smaller value of τ and a larger value of p in the ILUT implementation (see Table 6).

Poor preconditioning quality can also come from an unstable factorization, which is usually associated with small pivots encountered in the ILU factorizations [10,35]. Small pivots usually produce the L and U factors with large magnitude entries, which make the relatively small magnitude entries of the matrix A insignificant in finite precision computations. In sparse matrix computations, such problems can happen when performing ILU factorization on indefinite matrices [10,35].

In constructing an ILUT preconditioner for a dense matrix, we use single precision arithmetic to cope with the large amount of floating point operations and storage requirement. If the condition number of the matrix $M^{-1} = (LU)^{-1}$ is large, say, on the order of 10^7, computations in such a precision with the matrix M^{-1} is likely to be inaccurate. The preconditioned matrix $M^{-1}A$ might be worse conditioned than the original matrix A. In this situation, the preconditioned BiCG method may take more iterations to converge or may not converge at all. On the other hand, due to the implementation in which we stop the iteration when the (preconditioned) residual norm is reduced by 10^{-3}, it is possible that the initial residual norm is huge. Convergence might happen even if the actual residual norm is still large, if the initial residual norm is large. Such a convergence is called a *false* convergence.

To distinguish a false convergence from a true convergence, or to determine if a preconditioner might be effective before a preconditioned iteration procedure is carried out, it is informative to estimate the condition number of the $(LU)^{-1}$ matrix. Chow and Saad propose to estimate this number by calculating $||(LU)^{-1}e||_\infty$ where e is the vector of all ones. This statistics quantity is called *condest* in [10]. We note that this statistics is also a lower bound for $||(LU)^{-1}||_\infty$ and indicates a relation between the unstable triangular solutions and the poorly conditioned L and U factors [10,35]. The computation of the quantity $||(LU)^{-1}e||_\infty$ is just one forward solution step and one back substitution step with the L and U factors.

We calculate the condest numbers for the P1A case, in order to investigate what might happen with the ILUT factorization in solving a complex valued dense matrix. We find that the drop tolerance value τ (in the range of our studies) does not affect very much the value of condest. Condest value mostly depends on the fill-in parameter p. As the fill-in parameter p decreases and the drop tolerance parameter τ increases, the value of condest increases a lot in certain situations. We also find that the values of

Table 6
Condest values of the ILUT preconditioner for solving the P1A case

τ	p	Condest	it_{num}	Convergence	*sol*
10^{-6}	30	26.49327	39	Yes	same
	25	216.1559	64	Yes	same
	20	892906.2	77	Yes	diff
	15	7.83125E+09	2	Yes	diff
	10	3.17522E+19	–	Diverge	–
10^{-3}	30	26.43517	40	Yes	same
	25	188.4772	66	Yes	same
	20	192372.5	52	Yes	diff
	15	7.68935E+09	2	Yes	diff

condest are related to the solution patterns. If condest is very large, the preconditioned BiCG method does not converge within the given stopping criteria or diverges and the computed solution is not good, as we see in Table 6.

In Table 6, the ILUT with $p = 15$ and $\tau = 10^{-6}$ converges in 2 iterations, but the computed solution is not good. This is an example of a false convergence. We can determine the quality of this ILUT preconditioner without even starting the BiCG iteration process. The condest value of 7.83125×10^9 indicates that any single precision computation with $(LU)^{-1}$ is likely to be inaccurate.

6. Conclusions

In the implementation of the MLFMA, the ILU preconditioners based on a dual dropping strategy (ILUT) has been shown to reduce the number of the BiCG iterations dramatically, compared to the block diagonal preconditioner and without a preconditioner. Solving the large complex valued dense linear system arising from electromagnetic scattering using the BiCG method with an ILU preconditioner maintains the computational complexity of the MLFMA, and consequently reduces the total CPU time.

According to the results from our numerical experiments, we can see that the ILUT preconditioner constructed from the near part matrix improves the computational efficiency in the sense of both memory and CPU time. The results show that the BiCG method with the ILUT preconditioner is robust for solving 3D model cases from electromagnetic scattering simulations.

One advantage of our implementation is that we do not need access to the global matrix to construct our preconditioner (ideal for the MLFMA in which only A_{D} and A_{N} are numerically available). This is in contrast to many existing preconditioning techniques either in the incomplete LU factorization forms or in the sparse approximate inverse forms, in which a sparsified global matrix may be needed to construct a preconditioner [2,5].

We conducted a few experimental analysis in order to understand the properties of the ILUT preconditioned matrices. Our studies indicate that the eigenvalues of the ILUT preconditioned matrix are tightly clustered around 1, which ensures the fast convergence of the preconditioned BiCG method.

We discussed the situations that inaccurate or unstable ILUT preconditioners may be computed. We proposed to use a simple condest value to predict the quality of a constructed ILUT preconditioner before the preconditioned BiCG iteration is started. Our experimental results indicate that this strategy is practically useful in predicting false convergence and divergence of the preconditioned iterative solvers.

Appendix A. FMM formulation

The addition theorem for the 3D scalar Green's function is given by [9,11]

$$\frac{\mathrm{e}^{\mathrm{i}k_b|x+y|}}{|x+y|} = \int \mathrm{d}^2\hat{k}\, \mathrm{e}^{\mathrm{i}k_b\hat{k}\cdot x}\alpha(\hat{k},y), \quad |y| > |x|,$$

where

$$\alpha(\hat{k},y) = \frac{\mathrm{i}k_b}{4\pi}\sum_{l=0}^{L} \mathrm{i}^l(2l+1)h_l^{(1)}(k_b|y|)P_l(\hat{k}\cdot\hat{y}).$$

Let r and r' be the field point and the source point, respectively, the testing function f_j belongs to group-m centered at r_m, and the basis function f_i belongs to group-n centered at r_n. Let $x = (r - r_m) + (r_n - r')$, $y = (r_m - r_n)$, then $r - r' = x + y$. If group-m and group-n are well-separated, we have $|y| > |x|$, then

$$G(r,r') = \frac{\mathrm{e}^{\mathrm{i}k_b|r-r'|}}{4\pi|r-r'|} = \frac{1}{4\pi}\int \mathrm{d}^2\hat{k}\,\mathrm{e}^{\mathrm{i}k_b\hat{k}\cdot(r-r_m+r_n-r')}\alpha(\hat{k},r_{mn}),$$

where $r_{mn} = r_m - r_n$. Substituting this expansion into the expression for the matrix element, we have

$$Z_{ji} = \frac{\mathrm{i}\omega\mu_b}{4\pi}\int \mathrm{d}\hat{k}\,V_{fmj}(\hat{k})\cdot\alpha_{mn}(\hat{k})\cdot V_{sni}(\hat{k}),$$

where

$$V_{fmj}(\hat{k}) = (I - \hat{k}\hat{k})\int_{\Omega}\mathrm{d}\Omega f_j^{\Omega}(r)\,\mathrm{e}^{\mathrm{i}k_b\hat{k}\cdot(r-r_m)},$$

$$V_{sni}(\hat{k}) = (I - \hat{k}\hat{k})\int_{\Omega'}\mathrm{d}\Omega'\chi(r')f_i^{\Omega'}(r')\,\mathrm{e}^{-\mathrm{i}k_b\hat{k}\cdot(r'-r_n)}.$$

The matrix–vector multiplication for the nonnear part of $\sum_i Z_{ji}a_i$ can be written in a matrix format as $V_f\Lambda V_s$, which is the second part on the right-hand side of the Eq. (3).

References

[1] O. Axelsson, Iterative Solution Methods, Cambridge University Press, Cambridge, 1994.

[2] G. Alléon, M. Benzi, L. Giraud, Sparse approximate inverse preconditioning for dense linear systems arising in computational electromagnetics, Numer. Alg. 16 (1997) 1–15.

[3] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. van der Vorst, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, second ed., SIAM, Philadelphia, 1994.

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

[5] B. Carpentieri, I.S. Duff, L. Giraud, Experiments with sparse preconditioning of dense problems from electromagnetic applications, Technical Report TR/PA/00/04, CERFACS, Toulouse, France, 2000.

[6] B. Carpentieri, I.S. Duff, M. Magolu monga Made, Sparse symmetric preconditioners for dense linear systems in electromagnetism, Technical Report TR/PA/01/35, CERFACS, Toulouse, France, 2001.

[7] K. Chen, On a class of preconditioning methods for the dense linear systems from boundary elements, SIAM J. Sci. Comput. 20 (2) (1998) 684–698.

[8] E.F. D'Azevedo, F.A. Forsyth, W.P. Tang, Towards a cost effective ILU preconditioner with high level fill, BIT 33 (1990) 1–25.

[9] W.C. Chew, J.M. Jin, E. Midielssen, J.M. Song, Fast and Efficient Algorithms in Computational Electromagnetics, Artech House, Boston, 2001.

[10] E. Chow, Y. Saad, Experimental study of ILU preconditioners for indefinite matrices, J. Comput. Appl. Math. 86 (1997) 387–414.

[11] R. Coifman, V. Rokhlin, S. Wandzura, The fast multipole method for the wave equation: a pedestrian prescription, IEEE Antennas Propagat. Mag. 35 (3) (1993) 7–12.

[12] E. Darve, The fast multipole method: numerical implementation, J. Comput. Phys. 160 (2000) 195–240.

[13] A. Greenbaum, Iterative Methods for Solving Linear Systems, SIAM, Philadelphia, 1997.

[14] B.M. Kolundzija, Electromagnetic modeling of composite metallic and dielectric structures, IEEE Trans. Microwave Theory Tech. 47 (7) (1999) 1021–1032.

[15] C. Lanczos, Solution of systems of linear equations by minimized iterations, J. Res. Natl. Bur. Stand. 49 (1952) 33–53.

[16] J. Lee, J. Zhang, C. Lu, Incomplete LU preconditioning for large scale dense complex linear systems from electromagnetic wave scattering problems, Technical Report No. 342-02, Department of Computer Science, University of Kentucky, Lexington, KY, 2002.

[17] C.C. Lu, W.C. Chew, A multilevel algorithm for solving a boundary integral equation of wave scattering, IEEE Trans. Micro. Opt. Tech. Lett. 7 (10) (1994) 466–470.

[18] C.C. Lu, W.C. Chew, A coupled surface–volume integral equation approach for the calculation of electromagnetic scattering from composite metallic and material targets, IEEE Trans. Antennas Propagat. 48 (12) (2000) 1866–1868.

[19] J.A. Meijerink, H.A. van der Vorst, An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix, Math. Comput. 31 (1977) 148–162.

[20] J. Rahola, Experiments on iterative methods and the fast multipole method in electromagnetic scattering calculations, Technical Report TR/PA/98/49, CERFACS, Toulouse, France, 1998.
[21] S.M. Rao, D.R. Wilton, A.W. Glisson, Electromagnetic scattering by surface of arbitrary shape, IEEE Trans. Antennas Propagat. AP-30 (3) (1982) 409–418.
[22] V. Rokhlin, Rapid solution of integral equations of scattering theory in two dimensions, J. Comput. Phys. 86 (2) (1990) 414–439.
[23] Y. Saad, ILUT: a dual threshold incomplete LU factorization, Numer. Linear Algebra Appl. 1 (4) (1994) 387–402.
[24] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing Company, Boston, 1996.
[25] K. Sertel, J.L. Volakis, Incomplete LU preconditioner for FMM implementation, Micro. Opt. Tech. Lett. 26 (7) (2000) 265–267.
[26] T.K. Shark, S.M. Rao, A.R. Djordievic, Electromagnetic scattering and radiation from finite microstrip structures, IEEE Trans. Micro. Opt. Tech. 38 (11) (1990) 1568–1575.
[27] J.M. Song, W.C. Chew, Multilevel fast multipole algorithm for solving combined field integral equation of electromagnetic scattering, Micro. Opt. Tech. Lett. 10 (1) (1995) 14–19.
[28] J.M. Song, C.C. Lu, W.C. Chew, Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects, IEEE Trans. Antennas Propagat. AP-45 (10) (1997) 1488–1493.
[29] T. Vaupel, V. Hansen, Electrodynamic analysis of combined microstrip and coplanar/slotline structure with 3-D components based on a surface/volume integral equation approach, IEEE Trans. Microwave Theory Tech. 47 (9) (1999) 1150–1155.
[30] Y.V. Vorobyev, Method of Moments in Applied Mathematics, Gordon and Breach Science, New York, 1965.
[31] C.F. Wang, J.M. Jin, A fast full-wave analysis of scattering and radiation from large finite arrays of microstrip antennas, IEEE Trans. Antennas Propagat. 46 (10) (1998) 1467–1474.
[32] J.W. Watts III, A conjugate gradient truncated direct method for the iterative solution of the reservoir simulation pressure equation, Society of Petroleum Engineer J. 21 (1981) 345–353.
[33] J. Zhang, Preconditioned Krylov subspace methods for solving nonsymmetric matrices from CFD applications, Comput. Methods Appl. Mech. Engrg. 189 (3) (2000) 825–840.
[34] J. Zhang, Sparse approximate inverse and multilevel block ILU preconditioning techniques for general sparse matrices, Appl. Numer. Math. 35 (2000) 67–86.
[35] J. Zhang, A multilevel dual reordering strategy for robust incomplete LU factorization of indefinite matrices, SIAM J. Matrix Anal. Appl. 22 (3) (2001) 925–947.