Hindawi Publishing Corporation
International Journal of Antennas and Propagation
Volume 2012, Article ID 756259, 9 pages
doi:10.1155/2012/756259

Research Article

Hierarchical Matrices Method and Its Application in Electromagnetic Integral Equations

Han Guo, Jun Hu, Hanru Shao, and Zaiping Nie

School of Electronic Engineering, University of Electronic Science and Technology of China, Sichuan, Chengdu 611731, China

Correspondence should be addressed to Jun Hu, hujun@uestc.edu.cn

Received 15 July 2011; Revised 14 October 2011; Accepted 18 October 2011

Academic Editor: Ning Yuan

Hierarchical ($\mathcal{H}$-) matrices method is a general mathematical framework providing a highly compact representation and efficient numerical arithmetic. When applied in integral-equation- (IE-) based computational electromagnetics, $\mathcal{H}$-matrices can be regarded as a fast algorithm; therefore, both the CPU time and memory requirement are reduced significantly. Its kernel independent feature also makes it suitable for any kind of integral equation. To solve $\mathcal{H}$-matrices system, Krylov iteration methods can be employed with appropriate preconditioners, and direct solvers based on the hierarchical structure of $\mathcal{H}$-matrices are also available along with high efficiency and accuracy, which is a unique advantage compared to other fast algorithms. In this paper, a novel sparse approximate inverse (SAI) preconditioner in multilevel fashion is proposed to accelerate the convergence rate of Krylov iterations for solving $\mathcal{H}$-matrices system in electromagnetic applications, and a group of parallel fast direct solvers are developed for dealing with multiple right-hand-side cases. Finally, numerical experiments are given to demonstrate the advantages of the proposed multilevel preconditioner compared to conventional "single level" preconditioners and the practicability of the fast direct solvers for arbitrary complex structures.

1. Introduction

Integral equation (IE) method [1] is widely used in electromagnetic analysis and simulation. Compared to finite difference time domain (FDTD) method [2] and Finite element (FE) method [3], IE method is generally more accurate and avoids annoying numerical dispersion problems as well as complex boundary conditions. Despite the many advantages, IE method is often involved in enormous computational consumption, since its numerical discretization leads to dense linear system. With the efforts for years, various fast algorithms had developed focusing on reducing the computational complexity for IE method such as adaptive integral method (AIM) [4], fast multipole method (FMM) [5], IE-Fast Fourier transform (IE-FFT) [6], and fast low-rank compression methods [7], Though these fast algorithms are based on different theories, they use the same idea to reduce CPU time and memory usage complexity, which is to compute and store the major entries of the dense system matrix indirectly, and employ iterative methods instead of direct methods (such as LU decomposition) to solve the system. Preconditioners are often used to accelerate the convergence of iterative methods. However, iterative methods cannot always guarantee a reasonable solution with high precision; therefore, in some complex cases, we expect to employ powerful preconditioners to obtain visible acceleration of convergence or even use direct methods to avoid this problem entirely. But to implement these ideas in traditional fast algorithms encounters some difficulties that are, once a fast algorithm is applied, the major entries of the system matrix are computed and stored indirectly, and only a few entries can be accessed to form the sparse pattern which is essential to construct the preconditioners. This condition confines the flexibility of the preconditioning; therefore, a powerful preconditioner is generally hard to get. For direct methods, accessibleness of the whole matrix entries is also prerequisite, and employing them in fast algorithms is difficult as well as flexible preconditioning.

Hierarchical ($\mathcal{H}$-) matrices method [8, 9] is a general mathematical framework providing a highly compact representation and efficient numerical arithmetic. Applying $\mathcal{H}$-matrices in IE method can reduce both CPU time and

memory usage complexity significantly, so it can be regarded as a fast algorithm to IE method. Unlike the traditional fast algorithms mentioned above, any entry of $\mathcal{H}$-structured system matrix can be recovered easily, though the major entries are still implicitly expressed. This quality enables us to construct more efficient preconditioners, and moreover, a fast direct solver is available since its unique format. In this paper, a novel sparse approximate inverse (SAI) preconditioner in multilevel fashion is proposed to accelerate the convergence rate of Krylov iterations for solving $\mathcal{H}$-matrices system in electromagnetic applications, which is mentioned in [10], and a group of parallel fast direct solvers are developed for dealing with multiple right-hand-side cases.

This paper is organized as follows. Section 2 gives a brief review of the IE method and basic conception of $\mathcal{H}$-matrices. In Section 3, we elaborate the construction of the proposed multilevel SAI preconditioner and the implementation of parallel fast direct solvers. Numerical experiments are given in Section 4 to demonstrate the advantages of the proposed multilevel preconditioner compared to conventional "single level" preconditioners and the practicability of the fast direct solvers for arbitrary complex structures. Finally, some conclusions are given in Section 5.

2. $\mathcal{H}$-Matrices Representation for IE Method

We first proceed with a description of the IE method for solving electromagnetic scattering problems from 3D perfectly electric conductor (PEC). For concise introduction, only the electric field integral equation (EFIE) [1] is considered. The EFIE is written as

$$\hat{t} \cdot \int_{s'} ds' \overline{\mathbf{G}}(\mathbf{r}, \mathbf{r}') \mathbf{J}(\mathbf{r}') = \frac{4\pi i}{k\eta} \hat{t} \cdot \mathbf{E}^i(\mathbf{r}). \tag{1}$$

Discretize integral equation (1) by expanding $\mathbf{J}$ with local basis functions

$$\mathbf{J}(\mathbf{r}) = \sum_{n=1}^{N} x_n \mathbf{f}(\mathbf{r}), \tag{2}$$

where N is the number of unknowns, denoting the vector basis functions, and x_n is the unknown expansion coefficients. Applying Galerkin's method results in a matrix equation

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}, \tag{3}$$

where

$$\begin{aligned} \mathbf{A}_{mn} &= \int_s ds \mathbf{f}_m(\mathbf{r}) \cdot \int_{s'} ds' \overline{\mathbf{G}}(\mathbf{r}, \mathbf{r}') \cdot \mathbf{f}_n(\mathbf{r}'), \\ \mathbf{b}_m &= \frac{4\pi i}{k\eta} \int_s ds \mathbf{f}_m(\mathbf{r}) \cdot \mathbf{E}^i(\mathbf{r}). \end{aligned} \tag{4}$$

If matrix $\mathbf{A}$ is in $\mathcal{H}$-matrices formatted, we use $\mathbf{A}_{\mathcal{H}}$ to denote it.

Considering that the construction of proposed multilevel SAI preconditioner is built upon the structure of $\mathcal{H}$-matrices, some of its basic concepts need be reviewed at first. The basis functions set indexed by $\mathcal{I} := \{1, 2, \ldots, N\}$ can be constructed as a tree $T_{\mathcal{I}} = (V, E)$ with vertex set V and edge set E. For each vertex $v \in V$, we define the set of son of v as $S_{\mathcal{I}}(v) := \{w \in V \mid (v, w) \in E\}$. The tree $T_{\mathcal{I}}$ is called a cluster tree if the following conditions hold:

$$\begin{aligned} &\mathcal{I} \text{ is the root of } T_{\mathcal{I}}, \\ &\forall v \in V : \text{if } S_{\mathcal{I}}(v) \neq \emptyset, \text{we have } v = \bigcup_{w \in S_{\mathcal{I}}(v)} w, \\ &\forall w_1, w_2 \in S_{\mathcal{I}}(v) \quad \text{with } w_1 \neq w_2 : w_1 \cap w_2 = \emptyset. \end{aligned} \tag{5}$$

Each vertex v in $T_{\mathcal{I}}$ is called a cluster, representing a bunch of basis functions. Typically, each vertex has two sons. If the amount of basis functions contained in a cluster is less than a certain number of $n_{\min}$, that cluster has no sons, also called the leaves of $T_{\mathcal{I}}$. By rearranging the basis functions by their indices, they are numbered consecutively in each cluster, and, moreover, they are concentrated geometrically. Precisely, let Ω_v denote the domain of basis functions represented by cluster v, and it is bounded by the cardinality of v:

$$\operatorname{diam}(\Omega_v) \leq c_g |v| \tag{6}$$

in which $\operatorname{diam}(\cdot)$ is the Euclidean diameter of a set. c_g is a real constant; let the inequality valid for all clusters.

The electromagnetic interaction of any two clusters, including self-interaction, maps certain subblock of the system coefficient matrix. Practically, most of these subblocks can be approximated by low-rank matrices with high-accuracy. Therefore, a systemic and appropriate partitioning procedure is needed for the coefficient matrix. Based on the cluster tree $T_{\mathcal{I}}$ which contains hierarchy of partitions of $\mathcal{I}$, we are able to construct the block-cluster tree $T_{\mathcal{I}\times\mathcal{I}}$ describing a hierarchical partition of $\mathcal{I} \times \mathcal{I}$ by the following maps:

$$S_{\mathcal{I}\times\mathcal{I}}(t \times s) = \begin{cases} \emptyset, & \text{if } S_{\mathcal{I}}(t) = \emptyset \text{ or } S_{\mathcal{I}}(s) = \emptyset, \\ \emptyset, & \text{if } t \times s \text{ is admissible,} \\ S_{\mathcal{I}}(t) \times S_{\mathcal{I}}(s), & \text{otherwise,} \end{cases} \tag{7}$$

in which the definition of $S_{\mathcal{I}\times\mathcal{I}}(\cdot)$ is similar to $S_{\mathcal{I}}(\cdot)$ that denotes the sons of certain block-cluster $t \times s$. *Admissible* is a criterion which will be elaborated later and decides whether the subblock should be approximated by low-rank matrix, or split and combined by its sons, suspending for further transaction. Finally, the whole system matrix is segmented into pieces of subblocks by the procedure above. Corresponding to the block-cluster tree, these subblocks are called the leaves of $T_{\mathcal{I}\times\mathcal{I}}$, denoted by $\mathcal{L}(T_{\mathcal{I}\times\mathcal{I}})$.

If two clusters are well separated geometrically, the Green function which connects the interaction of them is barely varying in their domains. That means, only a few patterns of interactional vectors can represent the whole mode; therefore, the subblock representing their interaction is rank deficit. In order to discerning these subblocks appropriately, we introduce the admissibility condition [9]:

$$\max\{\operatorname{diam}(\Omega_t), \operatorname{diam}(\Omega_s)\} \leq \eta \operatorname{dist}(\Omega_t, \Omega_s), \tag{8}$$

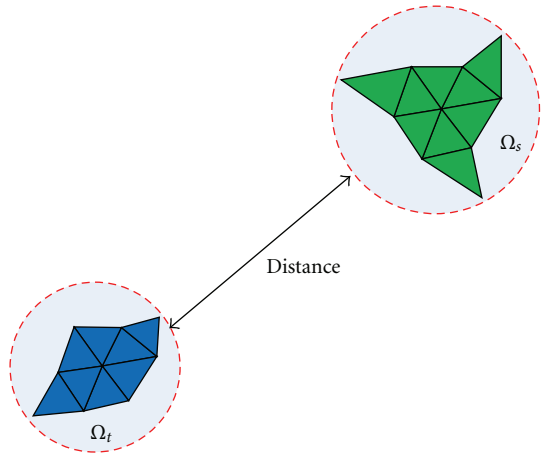


Figure 1: Two basis function clusters domains and their distance, describe the definition of *admissible* condition.

in which dist$(\cdot,\cdot)$ is the Euclidean distance of two sets, Ω_t and Ω_s denote the domains of basis functions of cluster t, and s and η are a controllable real parameter, which is also illustrated in Figure 1. If the statement (8) is true, sub-block $t \times s$ is *admissible*, otherwise it is *nonadmissible*. The leaves of block-cluster tree $\mathcal{L}(T_{\mathbb{1}\times\mathbb{1}})$ are either *admissible* or *nonadmissible*. For *admissible* subblocks, we substitute them by low-rank representation through specific algorithms such as adaptive cross approximation (ACA) [11]. For *nonadmissilbe* subblocks, we calculate the entries and store them directly. The total computational complexity for constructing is close to $O(N \log N)$, in which N regards to the number of unknowns, and we give a numerical investigation Part A of Section 4.

3. Multilevel SAI Preconditioning and Fast Direct Methods

3.1. Multilevel SAI Preconditioning. If the Krylov iterative methods are used to solve the linear system, we always expect to find a high-performance preconditioner to accelerate the convergence. Generally, instead of solving the linear system of the form $\mathbf{A}_{\mathcal{H}}\mathbf{x} = \mathbf{b}$, we solve it by the following forms:

$$\mathbf{M}\mathbf{A}_{\mathcal{H}}\mathbf{x} = \mathbf{M}\mathbf{b} \qquad \text{or } \mathbf{A}_{\mathcal{H}}\mathbf{M}\mathbf{y} = \mathbf{b}, \quad \mathbf{x} = \mathbf{M}\mathbf{y}. \tag{9}$$

In which $\mathbf{M}$ is a sparse matrix satisfying the condition $\mathbf{M} \approx \mathbf{A}_{\mathcal{H}}^{-1}$ to a certain degree. Therefore, $\mathbf{M}\mathbf{A}_{\mathcal{H}} \approx \mathbf{I}$ or $\mathbf{A}_{\mathcal{H}}\mathbf{M} \approx \mathbf{I}$ can make the iterative solver more efficient. Because of limited space, we only discuss the left preconditioning case below. The right preconditioning case can be analyzed similarly. Traditional preconditioner has a fixed form that we could only execute "single level" preconditioning. Here, we elaborate how to construct a multilevel preconditioner under $\mathcal{H}$-matrices format.

For a block-cluster tree $T_{\mathbb{1}\times\mathbb{1}}$ in $\mathcal{H}$-matrices, let $L(T_{\mathbb{1}\times\mathbb{1}})$ denotes its depth and $\mathcal{L}(T_{\mathbb{1}\times\mathbb{1}})$ denotes all the leaves of the cluster tree. Any leaf block-cluster $b = (t \times s) \in \mathcal{L}(T_{\mathbb{1}\times\mathbb{1}})$ belongs to certain level of $T_{\mathbb{1}\times\mathbb{1}}$; hence, $1 \le \text{level}(b) \le L(T_{\mathbb{1}\times\mathbb{1}})$. We define that he finest level is the 1st level, where the smallest block cluster is located. A substructure of $\mathbf{A}_{\mathcal{H}}$ denoted by $\mathbf{A}_{\mathcal{H}}^{(l)}$ can be defined as follows:

$$\mathbf{A}_{\mathcal{H}}^{(l)}(i,j) = \begin{cases} \mathbf{A}_{\mathcal{H}}(i,j), & (i,j) \in B^{(l)}, \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

in which $B^{(l)}$ is a subset of $\mathcal{L}(T_{\mathbb{1}\times\mathbb{1}})$,

$$B^{(l)} = \{b \mid b \in \mathcal{L}(T_{\mathbb{1}\times\mathbb{1}}), \text{level}\,(b) \le l\}, \tag{11}$$

so $\mathbf{A}_{\mathcal{H}}^{(l)}$ is sparse regarded as a standard matrix, which could be made use of as the primary data to form sparse approximate inverse. A trivial example of $\mathbf{A}_{\mathcal{H}}^{(l)}$ is that

$$\mathbf{A}_{\mathcal{H}}^{(L(T_{\mathbb{1}\times\mathbb{1}}))} = \mathbf{A}_{\mathcal{H}}. \tag{12}$$

The intuitive grasp of $\mathbf{A}_{\mathcal{H}}^{(l)}$ is shown in Figure 2. Consider an IE linear system matrix in $\mathcal{H}$-matrices form, the nonzero entries distribution of $\mathbf{A}_{\mathcal{H}}^{(1)}, \mathbf{A}_{\mathcal{H}}^{(2)}$, and $\mathbf{A}_{\mathcal{H}}^{(3)}$ are marked as shadow area corresponding to the whole matrix. Obviously high level of $\mathbf{A}_{\mathcal{H}}^{(l)}$ contains more useful data of the system.

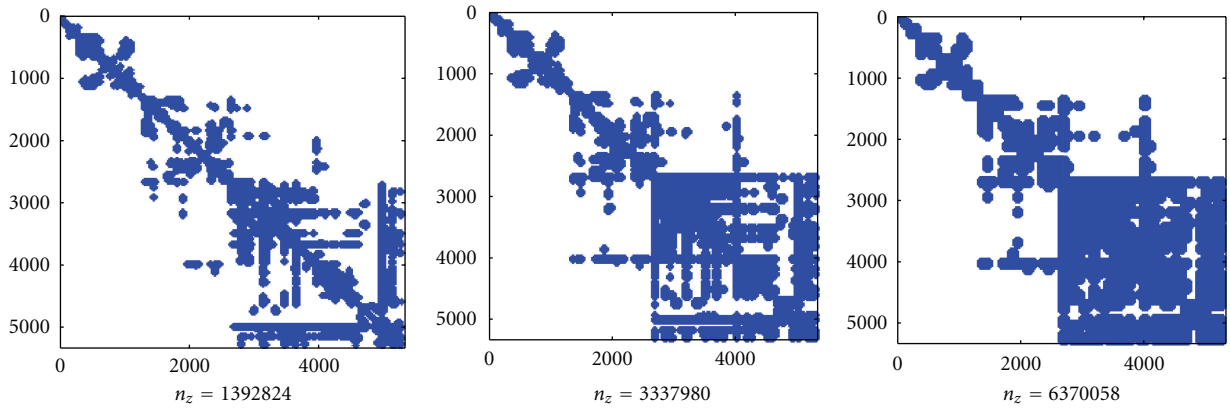
Correspondingly, a set of preconditioning matrices $\{M^{(l)} \mid l = 1, 2, \ldots L(T_{\mathbb{1}\times\mathbb{1}})\}$ can be generated through $\mathbf{A}_{\mathcal{H}}^{(l)}$, which satisfied the conditions below:

$$\begin{aligned} \left\|\mathbf{I} - \mathbf{M}^{(1)}\mathbf{A}_{\mathcal{H}}\right\| &\ge \left\|\mathbf{I} - \mathbf{M}^{(2)}\mathbf{A}_{\mathcal{H}}\right\| \\ &\ge \cdots \ge \left\|\mathbf{I} - \mathbf{M}^{(L(T_{\mathbb{1}\times\mathbb{1}}))}\mathbf{A}_{\mathcal{H}}\right\| \approx 0. \end{aligned} \tag{13}$$

In practice, $\mathbf{M}^{(l)}$ can be solved by the sparse approximate inverse technique, which aims to minimize $\|\mathbf{I} - \mathbf{M}^{(1)}\mathbf{A}_{\mathcal{H}}^{(l)}\|_F$ [12]. $\|\cdot\|_F$ is the Frobenius norm of a matrix, and $\mathbf{M}^{(l)}$ is subjected to a certain sparsity pattern, expressed as

$$\min_{\mathbf{M}^{(l)} \in G^{(l)}} \left\|\mathbf{I} - \mathbf{M}^{(l)}\mathbf{A}_{\mathcal{H}}^{(l)}\right\|_F^2 = \sum_{i=1}^{n} \min_{\mathbf{m}_i^{(l)} \in G_i^{(l)}} \left\|\mathbf{e}_i - \mathbf{m}_i^{(l)}\mathbf{A}_{\mathcal{H}}^{(l)}\right\|_2^2, \tag{14}$$

where $\mathbf{e}_i$ and $\mathbf{m}_i^{(l)}$ are the row vectors of the matrices $\mathbf{I}$ and $\mathbf{M}^{(l)}$, respectively. $G^{(l)}$ represents the sparsity pattern for



(a) Level 1 (b) Level 2 (c) Level 3

Figure 2: The data distribution of the finest 3 levels of $\mathbf{A}_{\mathcal{H}}^{(l)}$. This $\mathcal{H}$-matrices-formatted system is derived from an aircraft model which will be introduced in numerical experiments of Section 4.

different $\mathbf{M}^{(l)}$. Each of the subminimization problems in the right hand of (14) can be solved independently.

Because $\mathbf{m}_i^{(l)}$ in (14) is constrained to a certain sparse pattern G, it has many zero entries that force the corresponding rows of $\mathbf{A}_{\mathcal{H}}^{(l)}$ to be zero in matrix-vector multiply. Let $\widetilde{\mathbf{m}}_i^{(l)}$ denotes the subvector containing the nonzero entries of $\mathbf{m}_i^{(l)}$, and its corresponding rows of $\mathbf{A}_{\mathcal{H}}^{(l)}$ are denoted by $\mathbf{A}_{\mathcal{H}i}^{(l)}$. Besides, since $\mathbf{A}_{\mathcal{H}}^{(l)}$ is sparse, the submatrix $\mathbf{A}_{\mathcal{H}i}^{(l)}$ has many columns that are identically zero. By removing the zero columns, we have a much smaller submatrix $\widetilde{\mathbf{A}}_{\mathcal{H}i}^{(l)}$. The individual minimization problem of (14) is reduced to a least squares problem:

$$\min_{\widetilde{\mathbf{m}}_i^{(l)}} \left\| \widetilde{\mathbf{e}}_i - \widetilde{\mathbf{m}}_i^{(l)} \widetilde{\mathbf{A}}_{\mathcal{H}i}^{(l)} \right\|_2, \quad i = 1, 2, \ldots n. \tag{15}$$

$\mathbf{A}_{\mathcal{H}i}^{(l)}$ is just the submatrix of $\mathrm{A}_{\mathcal{H}}^{(l)}$ that we implement the QR decomposition for solving (15).

3.2. Hierarchical Fast Direct Methods. If the system matrix is severely ill-conditioned and even the iterative solver with powerful preconditioner cannot obtain acceptable results, fast direct solvers are good alternative for IE $\mathcal{H}$-matrices. According to the arithmetic of partition matrices, the inversion of a matrix can be calculated by operating its submatrices. Considering the hierarchical structure of $\mathcal{H}$-matrices is indeed a nested quaternary submatrices structure, and a potential direct method can be made to solve the linear system. If a system matrix $\mathbf{A}$ can be partitioned as

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}, \tag{16}$$

then its inversion can be written as

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{A}_{11}^{-1} + \mathbf{A}_{11}^{-1}\mathbf{A}_{12}\mathbf{C}^{-1}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} & -\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\mathbf{C}^{-1} \\ -\mathbf{C}^{-1}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} & \mathbf{C}^{-1} \end{bmatrix}, \tag{17}$$

Recursive procedure H_Inverse(m, x)
If the matrix m is partition **then**
  H_Inverse(m_{11}, x_{11})
  $x_{12} = -x_{11} \odot m_{12}$
  $x_{21} = -m_{12} \odot x_{11}$
  $m_{22} = m_{22} \oplus m_{21} \odot x_{12}$
  H_Inverse(m_{22}, x_{22})
  $m_{12} = x_{12} \odot m_{22}$
  $m_{11} = m_{11} \oplus m_{12} \odot x_{21}$
  $m_{21} = m_{22} \odot x_{21}$
else
  Direct_Inverse(m_{11})
end if

Algorithm 1

where $\mathbf{C} = \mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12}$. Applying this arithmetic into $\mathcal{H}$-matrices, we can obtain a hierarchical inversing procedure as elaborated in Algorithm 1.

The operator $\odot$ and $\oplus$ manipulate matrices multiplication and addition which are specific for $\mathcal{H}$-matrices. Reference [9] gives detailed information about these operators. By implementing submatrices partition, aggregation, and truncation of singular value decomposition (SVD), the hierarchical structure can be maintained after the manipulation of these operators, and, more importantly, both CPU time and memory cost are saved compared to conventional matrices arithmetic. Consequently, the complexity of hierarchical direct solvers is $O(N^2)$ for CPU time and $O(N^{1.5})$ for memory usage, in contrast with $O(N^3)$ and $O(N^2)$ of conventional direct solver. This leads to realizing kinds of fast direct solving processes including the hierarchical inversing algorithm described above.

For numerical implementation, hierarchical inversing is not as fast as hierarchical LU decomposition, which is another fast direct method based on matrices decomposition

[9, 13]. Considering LU decomposition for partition matrix **A**:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{L}_{11} & \\ \mathbf{L}_{21} & \mathbf{L}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{U}_{11} & \mathbf{U}_{12} \\ & \mathbf{U}_{22} \end{bmatrix} \tag{18}$$

matrices **L** and **U** can be figured out by the procedures:

(1) computing $\mathbf{L}_{11}$ and $\mathbf{U}_{11}$ from the LUD $\mathbf{L}_{11}\mathbf{U}_{11} = \mathbf{A}_{11}$;

(2) computing $\mathbf{U}_{12}$ from $\mathbf{L}_{11}\mathbf{U}_{12} = \mathbf{A}_{12}$;

(3) computng $\mathbf{L}_{21}$ from $\mathbf{L}_{21}\mathbf{U}_{11} = \mathbf{A}_{21}$;

(4) computing $\mathbf{L}_{22}$ and $\mathbf{U}_{22}$ from the LUD $\mathbf{L}_{22}\mathbf{U}_{22} = \mathbf{A}_{22} - \mathbf{L}_{21}\mathbf{U}_{12}$.

Process (2) and process (3) can be refined as subprocesses of partitioned LU decomposition; therefore, by recursive applying of this procedure, the whole decomposition can be achieved. In the process of recursion, when the partitioning reaches the finest level in which submatrices are explicit expressed, conventional LU decomposition is employed. After the decomposition done, the primary **L** and **U** are still in $\mathcal{H}$-form and overwrite the original $\mathbf{A}_{\mathcal{H}}$. Then, we can use partitioned backward/forward processes to solve the linear system which is similar to linear equation solving by conventional LU decomposition. For EFIE, we can utilize its symmetric property and then construct a fast LL^{T} decomposition solver which is even much faster and memory saving. Considering LL^{T} decomposition for partition matrix **A**:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{21}^{\mathrm{T}} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{L}_{11} & \\ \mathbf{L}_{21} & \mathbf{L}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{L}_{11}^{\mathrm{T}} & \mathbf{L}_{21}^{\mathrm{T}} \\ & \mathbf{L}_{22}^{\mathrm{T}} \end{bmatrix}, \tag{19}$$

matrices **L** can be figure out by the procedures:

(1) computing $\mathbf{L}_{11}$ from the LL^{T} $\mathbf{L}_{11}\mathbf{L}_{11}^{\mathrm{T}} = \mathbf{A}_{11}$;

(2) computng $\mathbf{L}_{21}$ from $\mathbf{L}_{21}\mathbf{L}_{11}^{\mathrm{T}} = \mathbf{A}_{21}$;

(3) computing $\mathbf{L}_{22}$ from the $\mathrm{LL}^{\mathrm{T}}\mathrm{D}$ $\mathbf{L}_{22}\mathbf{L}_{22}^{\mathrm{T}} = \mathbf{A}_{22} - \mathbf{L}_{21}\mathbf{L}_{21}^{\mathrm{T}}$.

Comparing with the procedure of hierarchical LU decomposition, one step is removed because we can obtain $\mathbf{L}_{12}^{\mathrm{T}}$ from trivial transposition. Therefore, the implementing of LL^{T} decomposition is approximately one time faster than that of LU decomposition, in a recursive way.

4. Numerical Experiments

4.1. Complexity of Constructing $\mathcal{H}$-Matrices. To investigate the complexity of constructing IE $\mathcal{H}$-matrices for electromagnetic analysis, there are two themes, namely, discretization density varying and wave frequencies varying. The former one means we change the number of unknowns along with varying the mesh density regarding electromagnetic wavelength. And the latter one means we change the number of unknowns along with varying wave frequencies, but the mesh density regarding wavelength is fixed. Here, we use a PEC sphere model of 1.0λ (wavelength) radius to test both these two themes, as shown in Figures 3 and 4.

From the investigation, we can easily see that if incident wave frequency is fixed, the complexity of both CPU time and memory usage approach to $O(N \log N)$, where N regards to the number of unknowns. If we fix the discretization density and change the frequency of incident wave, the complexity curve is close to $O(N^{4/3} \log N)$. This is because the object domain contains more phase information when incident by higher frequency wave; therefore, the compression ratio of low rank submatrices in $\mathcal{H}$-matrices is lower in higher frequency cases.

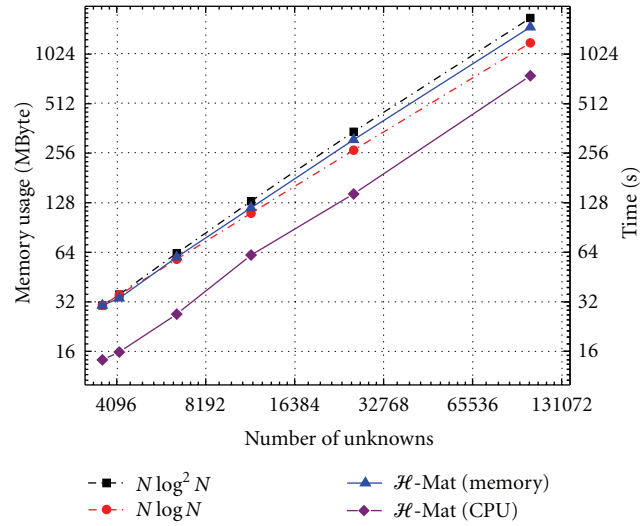


Figure 3: Computational complexity of constructing $\mathcal{H}$-matrices by fixed incident wave at 300.0 MHz.

4.2. Iterative Solving with Multilevel SAI Preconditioner. Firstly, a conducting sphere is used to demonstrate the improvement of the spectrum characteristics of the linear coefficient matrix by employing multilevel-SAI preconditioner (ML-SAI). Supposing $\mathbf{M}^{(l)}$ is the lth level preconditioner and $\mathbf{x}_{\mathrm{rnd}}^{(i)}$ is the ith randomized vector, we define the regression index $c_r^{(p)}$ of lth preconditioning level as

$$c_r^{(p)}(l) = \frac{1}{p}\sum_{i=1}^{p} \frac{\left\| \mathbf{x}_{\mathrm{rnd}}^{(i)} - \mathbf{M}^{(l)}\mathbf{A}_{\mathcal{H}}\mathbf{x}_{\mathrm{rnd}}^{(i)} \right\|_2}{\left\| \mathbf{x}_{\mathrm{rnd}}^{(i)} \right\|_2}, \tag{20}$$

if $\mathbf{M}^{(l)}\mathbf{A}_{\mathcal{H}}$ is close to identity matrix $\mathbf{I}$, then $c_r^{(p)}(l) \to 0$ is expected. Particularly, $\mathbf{M}^{(0)} = \mathbf{I}$, which means no preconditioner is imposed. A sphere of 2λ diameter, discretized by Rao-Wilton-Gliso (RWG) basis with different densities, is used to obtain the linear system matrix $\mathbf{A}_{\mathcal{H}}$. The different regression indices of different preconditioning levels are presented in Table 1 and compared to analogous indices of conventional SAI preconditioning in MLFMA.

From Table 1, we can see that with the preconditioning level of ML-SAI increased, the regression index is decreased, which means the effect of preconditioning is getting better. And the preconditioning effect of conventional SAI in MLFMA is better than the low level ML-SAI but not as good as high level ML-SAI. It should be noted that the $c_r^{(100)}(3)$ in case of 957 unknowns is close to zero because the $\mathcal{H}$-matrices

Table 1: The regression index of different levels of ML-SAI and conventional SAI preconditioning.

	Regression index	Number of unknowns			
		957	3,972	16,473	65,892
$\mathcal{H}$-Mat.	$c_r^{(100)}(0)$	5.568	2.297	1.371	1.042
	$c_r^{(100)}(1)$	0.1311	0.2440	0.3155	0.4237
	$c_r^{(100)}(2)$	0.08925	0.2034	0.2523	0.2816
	$c_r^{(100)}(3)$	1.116*E*-6	0.1112	0.1828	0.2165
ML-FMA	None	5.723	2.415	1.355	0.989
	SAI	0.1052	0.2237	0.2844	0.3125

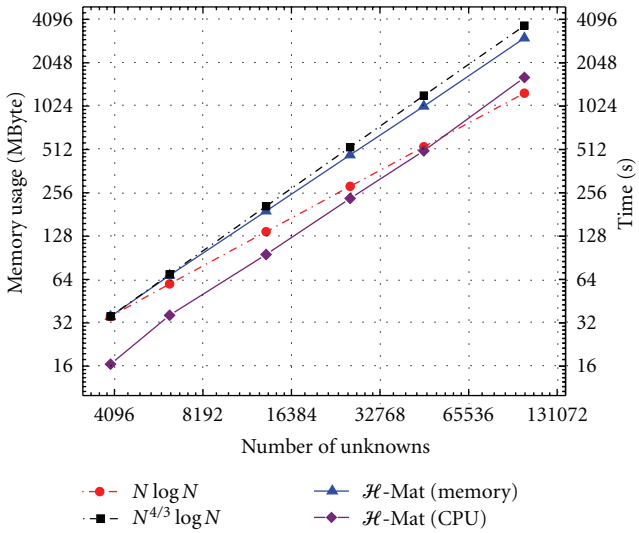


Figure 4: Computational complexity of constructing $\mathcal{H}$-matrices by fixed discretization mesh density at 8 points per wavelength.

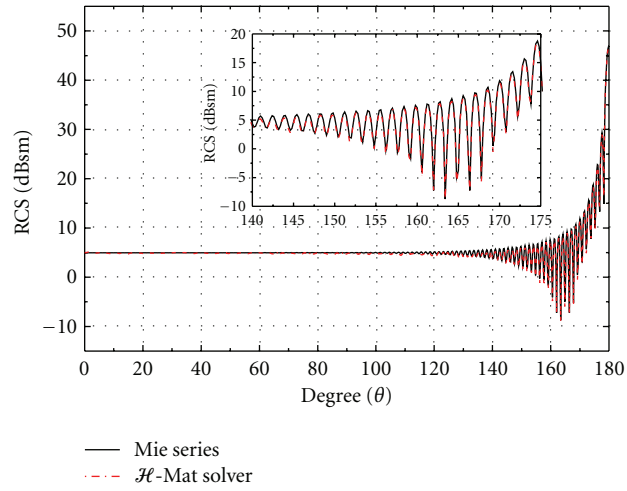


Figure 5: The bistatic RCS of a conducting sphere with 40 λ.

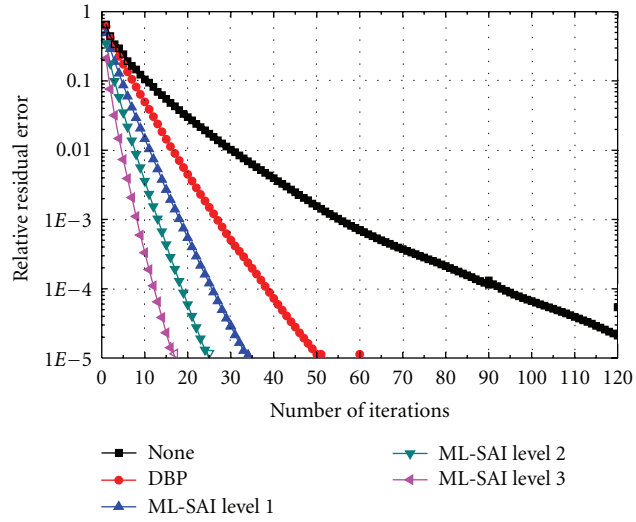


Figure 6: The iterative history of solving the PEC sphere cases. GMRES(30) is used and accelerated by different preconditioners.

structure of this model is only 3 levels, which means $\mathbf{M}^{(3)}$ is very close to the exact inverse of the linear system.

Next, a 40 λ diameter PEC sphere is solved by $\mathcal{H}$-matrices method. Combine field integral equation (CFIE) is used, and the number of unknowns is 426,827. To set up the hierarchical system matrix, the memory cost is 42.6 GBytes and 26,387 seconds are used by 8 core parallel computing, on a workstation of Intel Xeon X5460 processor with 64 GBytes RAM. Generalized minimal residual (GMRES) [14] algorithm is employed as the Krylov iterative solver, and we use 1~3 level SAI preconditioners and diagonal block preconditioners (DBPs) as reference to accelerate the convergence of iteration. From Figure 5, we can see that the RCS curve conform to the Mie series identically. And the iterative history in Figure 6 shows that ML-SAIs have a distinct impact on accelerating the convergence of iteration.

Another example is an aircraft model which is 50.75 λ long, 29.20 λ wide, and 13.57 λ high, excited by a plane wave incoming from its nose with an upper 45° angle to the center axis of its main body. The CFIE is used, and there are 412,737 unknowns to simulate the exciting surface currents as shown in Figure 7. By solving the linear system by $\mathcal{H}$-matrices method, there are 14 levels in the block-cluster tree. To set up the hierarchical system matrix, the memory cost is 38.4 GBytes and 25,634 seconds are used by 8 core parallel computing, on a workstation of Intel Xeon X5460 processor with 64 GBytes RAM. The data of $\mathbf{A}_{\mathcal{H}}^{(l)}$ in the finest 3 levels are to construct preconditioners $\mathbf{M}^{(1)}$, $\mathbf{M}^{(2)}$, and $\mathbf{M}^{(3)}$, respectively.

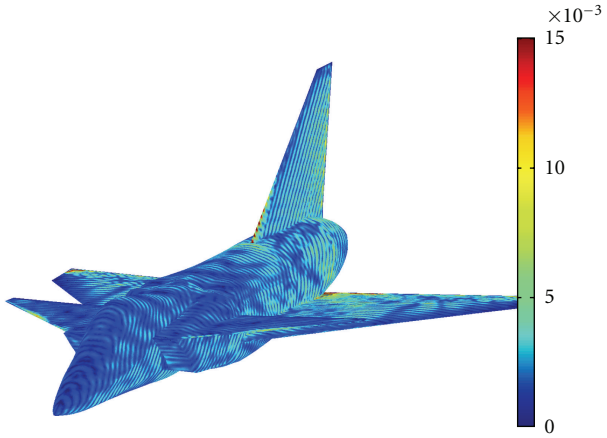


Figure 7: The surface current distribution of the aircraft model obtained by solving the ML-SAI preconditioned $\mathcal{H}$-matrices method. The unit of surface current is A/m^2.

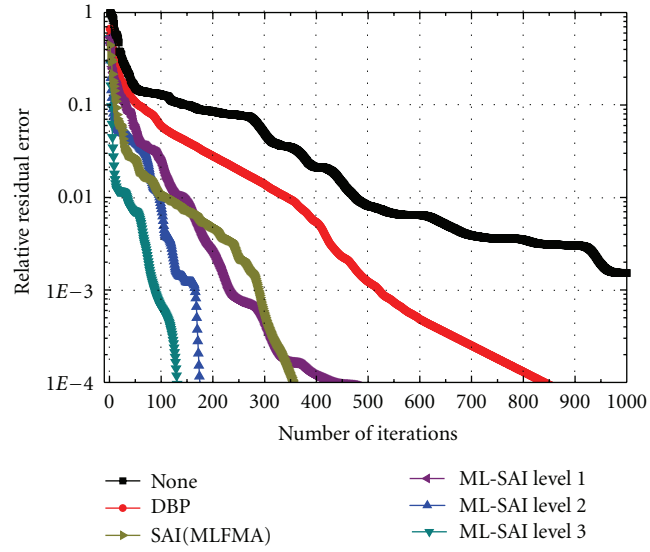


Figure 8: The convergence histories of GMRES(90) with no preconditioner, DB preconditioner, and ML-SAI preconditioners of different levels.

GMRES(90) is used to solve the linear system. The convergence histories of iteration with different preconditioners are presented in Figure 8. This is a little tough case because of complicated structure and relatively high frequency. If no preconditioner or just DBP is employed, the convergence is very poor. By employing ML-SAI, the iteration converges quickly under satisfied residual error, and the convergence is much better with the higher level preconditioning processed. The conventional SAI with MLFMA is also able to achieve the request, but its convergence is not as good as higher level ML-SAI. Table 2 shows the solving time and memory cost of ML-SAI, conventional SAI with MLFMA, and diagonal blocks preconditioner. The memory cost is mainly constituted by the storage of preconditioner **M** and Krylov subspace vectors of GMRES. In the fact that the restart number of GMRES, namely, the number of subspace vectors, is chosen the same for all cases, so the memory cost can be regarded as the scale of preconditioner **M**.

Table 2: The time and memory cost of iterative solving aircraft model with different preconditioners.

	Time (seconds)	Memory (MBytes)
ML-SAI level 1	582.8	325.4
ML-SAI level 2	234.5	464.5
ML-SAI level 3	184.3	786.5
SAI with MLFMA	643.7	312.3
DBP	1134.2	213.9

4.3. Hierarchical Fast Direct Solvers. In this part, we give some numerical examples solved by fast $\mathcal{H}$-matrices direct solvers. We use hierarchical LU decomposition to solve CFIE and use hierarchical LL^T decomposition to solve EFIE, since utilizing its symmetric property. First, a PEC sphere model is used to investigate the accuracy of fast direct solvers. From Figure 9, we can see that the bistatic RCSs obtained by both fast LU and LL^T decomposition solvers have very high precision regarding to analytic Mie series. The EFIE result has a slight bigger relative error than CFIE because the system matrix of EFIE is not diagonal dominated, consequently, the numerical error accumulated through LL^T decomposition is more than that in CFIE.

Next, the computational complexity of fast LL^T decomposition solver is tested via the PEC sphere model. The number of unknowns is varying along with the change of frequency of the incident wave, and the mesh discretization density is fixed at 10.0 points per wavelength. The test results are shown in Figures 10 and 11, respectively, in which we can observe that the complexity of CPU time is close to $O(N^2)$ and memory usage is close to $O(N^{1.5})$. This is much more efficient than conventional direct solvers.

The aircraft model shown in the previous part is also used here to test fast LL^T decomposition solver by employing EFIE. The incident wave is 10.0 GHz, and mesh discretization density is 10.0 points per wave number. The number of unknowns is 119,202, and 8 levels are set for $\mathcal{H}$-matrices. This example is parallelly solved by a workstation equipped with dual Intel Xeon X5460 processor and 64 GBytes of RAM. 2,256 seconds is used for building up the hierarchical system matrix, and the memory cost is 5,128.4 MBytes. LL^T decomposition time is 8,415 seconds, and 6,437.2 MBytes are used after decomposition. Figure 12 presents the surface current solved by LL^T decomposition.

The most notable advantage of fast direct solver is the high efficiency of handling multiple right-hand-side cases. After LU or LL^T decomposition, solving multiple right-hand vectors is only one-round process of forward/backward substitution, so that it is much faster than iterative solvers, which gives a full-round iteration for each vector step by step. In IE electromagnetic analysis, monostatic RCS is the typical multiple right-hand-side problem. We here use LL^T decomposition to solve the monostatic RCS problem of the aircraft model incident by 3.0 GHz electromagnetic wave. The range of sweep angles is 180°, and 181 samples are

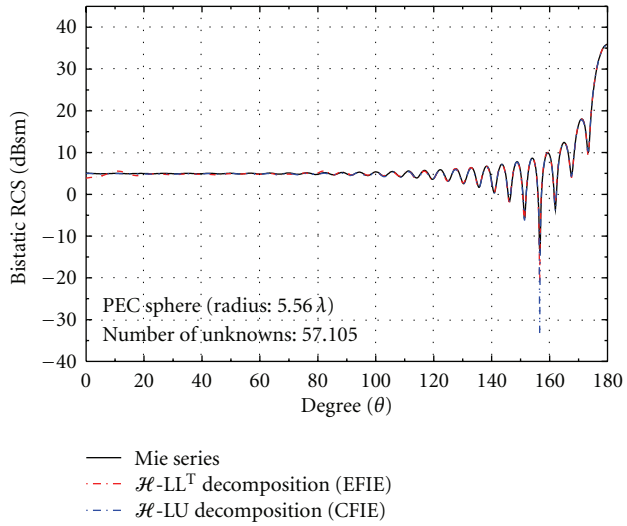


FIGURE 9: The bistatic RCS solved by hierarchical fast direct solvers. CFIE is solved by LU decomposition solver, and EFIE is solved by LL$^{\mathrm{T}}$ decomposition.

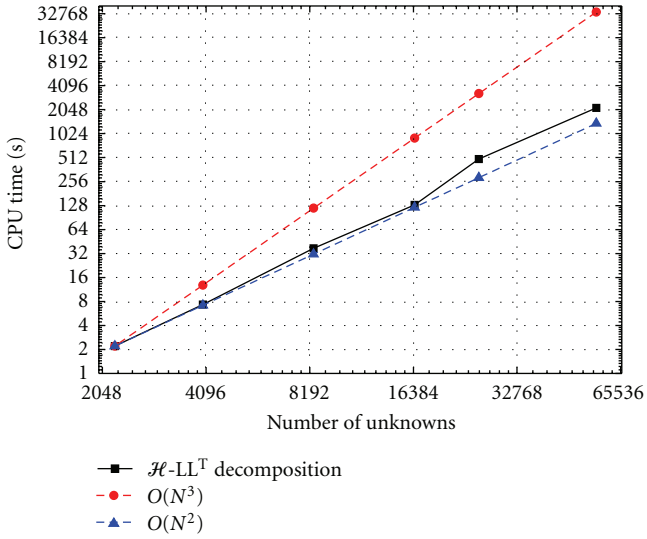


FIGURE 10: CPU time complexity of hierarchical LL$^{\mathrm{T}}$ decomposition.

calculated. The total solving time is 468.3 seconds, contrast to MLFMA combined with GMRES iterative solver, which costs 4215.7 seconds, almost ten times of the former one. In order to compare to LL$^{\mathrm{T}}$ decomposition, here, we use EFIE for iterative MLFMA. Figure 13 shows the comparing results between LL$^{\mathrm{T}}$ decomposition and iteration of MLFMA, we can observe that direct solved curve is smoother than that of iterative solved and there is a big difference between them. We believe that the result of LL$^{\mathrm{T}}$ decomposition is more accurate and the reason is elaborated as follow. Theoretically, the monostatic RCS curve of this intermediate-frequency example should be smooth, because the sampling is sufficiently continuous comparing to the wavelength. However, for EFIE iteration, the convergence is poor due to

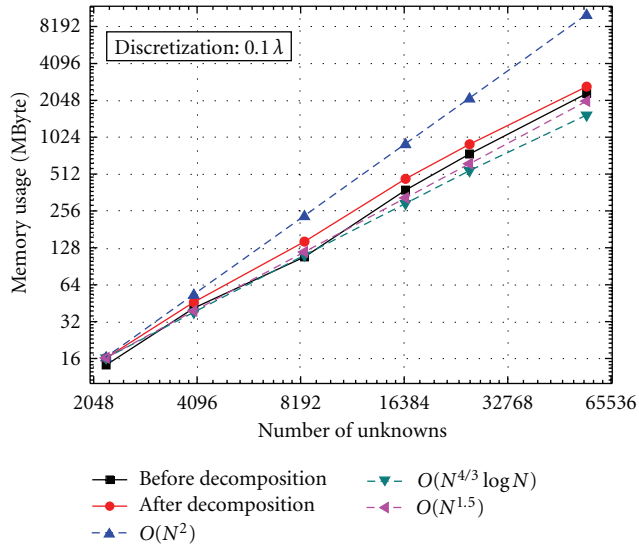


FIGURE 11: Memory usage complexity of hierarchical LL$^{\mathrm{T}}$ decomposition.

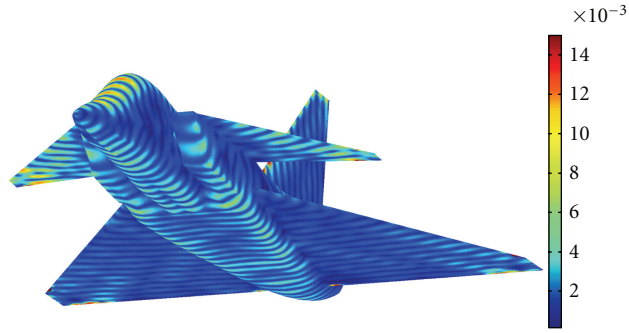


FIGURE 12: The surface current distribution of the aircraft model obtained by solving EFIE with hierarchical LL$^{\mathrm{T}}$ decomposition. The frequency of incident wave is at 10.0 GHz. The unit of surface current is A/m^2.

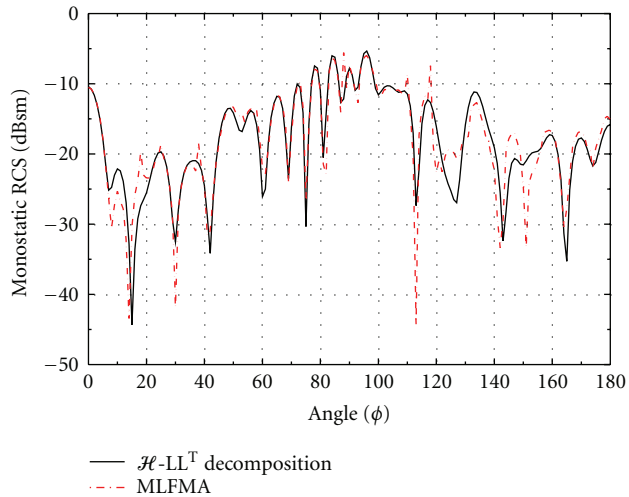


FIGURE 13: The monostatic RCS of the aircraft model. The incident wave is 3.0 GHz, HH-polarization.

the large condition number of its system matrix. Therefore, a loose relative residual error threshold 0.005 is set here, for the purpose of getting iterative result with a reasonable CPU time cost, and this causes the iterative result not very accurate.

5. Conclusion

The hierarchical matrice methods presented in this paper is embedded in electromagnetic IE method. Due to its special structure, we can construct a multilevel SAI preconditioner to accelerate the convergence of iterative solving, and even kinds of fast direct solvers can be made, which is not viable for the traditional IE fast algorithm. The multilevel SAI preconditioner proposed here is more efficient than conventional "single level" preconditioners, and hierarchical fast direct solvers are good alternatives to iterative solvers, very suitable for ill-conditioned system and multiple right-hand-side problems. Furthermore, the kernel independence feature of hierarchical matrices method is adapted to varied electromagnetic problems without being limited to integral equation with free-space Green function.

Acknowledgments

This work is supported by the Fundamental Science Research Foundation of National Central University for Doctoral Program (E022050205) and partly supported by NSFC (no. 60971032), the Programme of Introducing Talents of Discipline to Universities under Grant b07046.

References

[1] R. F. Harrington, *Field Computation by Moment Methods*, MacMillan, New York, NY, USA, 1968.

[2] A. Taflve, *Computational Electrodynamics: The Finite-Difference Time-Domain Method*, Artech House, Norwood, Mass, USA, 1995.

[3] J.-M. Jin, *The Finite Element Method in Electromagnetics*, Wiley, New York, NY, USA, 2002.

[4] E. Bleszynski, M. Bleszynski, and T. Jaroszewicz, "AIM: adaptive integral method for solving large-scale electromagnetic scattering and radiation problems," *Radio Science*, vol. 31, no. 5, pp. 1225–1251, 1996.

[5] J. Hu, Z. P. Nie, J. Wang, G. X. Zou, and J. Hu, "Multilevel fast multipole algorithm for solving scattering from 3-D electrically large object," *Chinese Journal of Radio Science*, vol. 19, no. 5, pp. 509–524, 2004.

[6] S. M. Seo and J. F. Lee, "A fast IE-FFT algorithm for solving PEC scattering problems," *IEEE Transactions on Magnetics*, vol. 41, no. 5, pp. 1476–1479, 2005.

[7] K. Zhao, M. N. Vouvakis, and J. F. Lee, "The adaptive cross approximation algorithm for accelerated method of moments computations of EMC problems," *IEEE Transactions on Electromagnetic Compatibility*, vol. 47, no. 4, pp. 763–773, 2005.

[8] W. Hackbusch, "Sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: introduction to $\mathcal{H}$-matrices," *Computing*, vol. 62, no. 2, pp. 89–108, 1999.

[9] S. Borm, L. Grasedyck, and W. Hackbusch, "Hierarchical matrices," *Lecture Note 21 of the Max Planck Institute for Mathematics in the Sciences*, 2003.

[10] H. Guo, J. Hu, H. Shao, and Z. Nie, "Multilevel sparse approximate inverse preconditioning for solving dynamic integral equation by H-matrix method," in *Proceedings of the IEEE International Workshop on Antenna Technology: Small Antennas, Novel Structures and Innovative Metamaterials*, pp. 124–127, Hong Kong, 2011.

[11] M. Bebendorf, "Approximation of boundary element matrices," *Numerische Mathematik*, vol. 86, no. 4, pp. 565–589, 2000.

[12] J. Lee, J. Zhang, and C. C. Lu, "Sparse inverse preconditioning of multilevel fast multipole algorithm for hybrid integral equations in electromagnetics," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 9, pp. 2277–2287, 2004.

[13] M. Bebendorf, *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*, Springer, Berlin, Germany, 2008.

[14] L. Y. Saad and M. H. Schultz, "GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 3, pp. 856–869, 1986.