

# An Effective Sparse Approximate Inverse Preconditioner for the MLFMA Solution of the Volume-Surface Integral Equation

Jinbo Liu<sup>1</sup>, Zengrui Li<sup>1</sup>, Mang He<sup>2</sup>, and Jianxun Su<sup>1</sup>

<sup>1</sup>School of Information and Communication Engineering  
Communication University of China, Beijing, 100024, P. R. China  
liuj@cuc.edu.cn, zrli@cuc.edu.cn, sujianxun\_jlgx@163.com

<sup>2</sup>School of Information and Electronics  
Beijing Institute of Technology, Beijing, 100081, P. R. China  
hemang@bit.edu.cn

**Abstract** — In the framework of the multilevel fast multipole algorithm (MLFMA), effective construction of the sparse approximate inverse preconditioner (SAIP) for the volume-surface integral equation (VSIE) is discussed. A high quality SAIP for the entire VSIE matrix is constructed by using the sub-matrix of the near-field interactions between the surface basis and testing functions arising from the surface integral equation alone. In addition, a simple sparse pattern selection scheme based on the geometrical information of nearby basis functions and octree regrouping strategy is proposed to enhance the efficiency of the SAIP. In contrast to the existing sparse pattern selection schemes, the proposed scheme utilizes the near-field matrix in the MLFMA more effectively with only one tuning parameter. Numerical results indicate that with the proposed scheme, both the memory usage and setup time for constructing an effective SAIP are significantly reduced without compromising the efficiency and robustness.

**Index Terms** — Method of moments (MoM), multilevel fast multipole algorithm (MLFMA), sparse approximate inverse preconditioner, volume-surface integral equation (VSIE).

## I. INTRODUCTION

The volume-surface integral equation (VSIE) [1], in conjunction of the method of moments (MoM) [2], is one of the most attractive methods to calculate the electromagnetic (EM) scattering or radiation of composite objects involving both conductors and inhomogeneous dielectrics. For the objects with electrically large sizes, fast EM algorithms, such as the multilevel fast multipole algorithm (MLFMA) [1], are highly required to alleviate the computation overhead. Based on the addition theorem of Green's function and

diagonalization of the translation operator, the MLFMA drastically reduces the overall computational complexity from the order of  $O(N^2)$  to  $O(M \log N)$ , where  $N$  is the number of unknowns [1]. During the implementation of MLFMA, the MoM matrix equation is decomposed into two parts as:

$$Z_{near}I + Z_{far}I = V, \quad (1)$$

where  $Z_{near}$  and  $Z_{far}$  are the  $N \times N$  impedance matrices representing the reactions between the basis and testing functions in the neighbor and far leaf boxes at the finest level of the MLFMA, and  $I$  and  $V$  are the  $N \times 1$  vectors of unknown expansion coefficients and generalized voltage, respectively. Besides, only the near-field matrix  $Z_{near}$  is explicitly computed and stored, while the far-field interaction part  $Z_{far}I$  is implicitly computed through three processes: aggregation, translation, and disaggregation.

Although the VSIE is a second-kind integral equation, it is still necessary to apply the preconditioning techniques to speed up the convergence during the iterative solution of the matrix equation. Among related preconditioning techniques, the sparse approximate inverse preconditioner (SAIP), which is based on directly approximating the inverse of the preconditioning matrix, has been proved that it is not only one of the most effective preconditioners for the surface integral equation (SIE), but also easy for parallelization [3-6]. In the MLFMA, utilizing  $Z_{near}$  can provide effective SAIP. Further, benefited from the octree structure of the MLFMA, the setup time for constructing the SAIP can be decreased substantially [3, 4]. But meanwhile, the construction process of SAIP is very sensitive to the average number of unknowns belonging to the leaf boxes. Especially for the VSIE, because of the three-dimensional volumetric unknowns which are densely distributed in each non-empty leaf box, the setup time for SAIP will be very long. To reduce the time cost, several sparse pattern selection schemes have been reported. In

[5], performances of various selection schemes based on algebraic or geometrical information were compared through a large set of numerical experiments, while it was concluded that the ones based on geometrical information of the basis function distributions are more effective.

However, most of the existing discussions on the SAIP were focused on its applicability in the solution of SIE, while its effectiveness on the VSIE is rarely studied. In this paper, how to effectively use the SAIP in the VSIE solution is investigated. Based on the geometrical information of nearby basis functions and octree regrouping strategy, an effective sparse pattern selection scheme for the SAIP construction is proposed. Compared to the conventional selection schemes, the proposed one is more effective and robust with only one tuning parameter.

## II. SAIP FOR THE VSIE

In the MLFMA, only the near-interaction matrix  $Z_{near}$  needs to be stored explicitly [1]. If we use  $\bar{Z}$  to denote the preconditioning matrix extracted from  $Z_{near}$  to construct the SAIP, then the preconditioner  $M$  can be constructed as an approximate inverse matrix of  $\bar{Z}$ . During the construction process, the Frobenius norm minimization is usually chosen since it can decouple the entire constrained minimization problem into  $N$  independent linear least-square problems as:

$$\min \|E - \bar{Z}M\|_F^2 = \min \sum_{k=1}^N \|e_k - \bar{Z}m_k\|_2^2, \quad (2)$$

which can be parallelized naturally. In (2),  $E$  is the identity matrix, and  $e_k$  and  $m_k$  are the  $k$ th column of the matrices  $E$  and  $M$ , respectively, while  $M$  is constrained by a certain sparse pattern. Since  $\bar{Z}$  is sparse, as done in [3, 4], let  $J$  denote the column structure of nonzero elements of  $\bar{Z}$ , and  $I$  denote the set of row indices of the nonzero entries of  $\bar{Z}(:, J)$ , the least-square problems in (2) are reduced into:

$$\begin{aligned} \min \sum_{k=1}^N \|e_k - \bar{Z}m_k\|_2^2 &\Rightarrow \\ \min \sum_{k=1}^N \|e_k(I) - \bar{Z}(I, J)m_k(J)\|_2^2 &, \end{aligned} \quad (3)$$

which can be efficiently solved by QR factorization, and the computational complexity is determined by the size of  $\bar{Z}(I, J)$ . The QR factorization requires asymptotically  $n_1 n_2^2$  flops, where  $n_1$  and  $n_2$  are the number of elements in the sets  $I$  and  $J$ , respectively. If we assume that each row of  $\bar{Z}$  has  $n$  nonzero entries in average, both  $n_1$  and  $n_2$  are the order of  $O(n)$ . Thus, the total computational complexity of (3) is in the order of  $O(Nn^3)$ . In the context of the MLFMA, when we choose  $\bar{Z}=Z_{near}$ , it is found that any testing function in a given leaf box will couple with the same set of basis functions located in the boxes of the

near-interaction list of the given box [3]. By using this fact, the computation overhead of (3) can be reduced to  $O(N_g n^3)$ , where  $N_g$  is the number of non-empty leaf boxes. Therefore, the setup time for constructing  $M$  can be reduced several-fold without any loss of accuracy. On the other hand,  $O(N_g n^3)$  indicates that the SAIP setup time is very sensitive to  $n$ . In other words, when  $n$  is relatively large, the setup time for constructing  $M$  will be very high. To alleviate this problem, after careful investigation, we find that there are two possible ways to reduce the SAIP computational effort for the VSIE.

The first way is to choose appropriate sub-matrices to construct the SAIP. In the VSIE, particularly,  $Z_{near}$  can be represented by four sparse sub-matrices as:

$$Z_{near} = \begin{bmatrix} Z_{SS} & Z_{SV} \\ Z_{VS} & Z_{VV} \end{bmatrix}, \quad (4)$$

where  $Z_{SS}$ ,  $Z_{SV}$ ,  $Z_{VS}$  and  $Z_{VV}$  denote the self/mutual interactions between two types of basis/testing functions in the near-interaction lists ( $S$  stands for surface functions and  $V$  for the volume ones). Because the SAIP can be applied to any square matrix, there are three optional schemes to construct the preconditioner  $M$ : using  $Z_{SS}$  only (denoted by  $SS$ ), using both  $Z_{SS}$  and  $Z_{VV}$  ( $SS\&VV$ ), and using the entire  $Z_{near}$  matrix ( $Tot$ ). For composite objects involving open conducting structures,  $Z_{SS}$  stands for the interactions between surface basis and testing functions used in the electric field integral equation (EFIE) which is a first-kind of integral equation, while  $Z_{VV}$  represents the interactions between volume basis and testing functions for the volume integral equation (VIE) that is second-kind. In general,  $Z_{VV}$  is more well-conditioned than  $Z_{SS}$ . Therefore, it is reasonable that applying SAIP to the sub-matrix  $Z_{SS}$  alone to improve the condition number of  $Z_{SS}$  might be enough for the calculation of composite objects. Besides, compared with  $SS\&VV$  and  $Tot$ ,  $SS$  is apparently the most memory- and time-efficient for constructing  $M$ . During the implementation of  $SS$ ,  $M$  is considered as:

$$M = \begin{bmatrix} M_{SS} & \\ & 1 \end{bmatrix}, \quad (5)$$

where  $M_{SS}$  is an approximate inverse matrix of  $Z_{SS}$  as:

$$Z_{SS}M_{SS} \approx E, \quad (6)$$

which can be efficiently solved using (2).

On the other hand, the objects including thin conducting structures are needed to be discretized by fine meshes. If  $\bar{Z}$  is still chosen the same as  $Z_{near}$ ,  $\bar{Z}$  will be very dense, resulting in a large  $n$  as well as high computation complexity of (3). In order to overcome this difficulty, it is necessary to select effectual sparse pattern from  $Z_{near}$ , i.e., extracting a sparser matrix  $\bar{Z}$  from  $Z_{near}$  by pre-filtering technique before constructing  $M$ , which can evidently reduce the value of  $n$ . There are two main types of schemes reported to achieve this goal: 1) dropping entries with tiny magnitudes in  $Z_{near}$  as done in

[6]. However, improper setting of the dropping threshold may even degrade the convergence [5]. Thus, this scheme is not widely used; 2) dropping entries based on the geometrical information during the MLFMA implementation as done in [3]. Specifically, for a given testing function  $X_k$  located in the leaf box  $G$ , when drops the entries from  $Z_{near}$  to reduce into  $\bar{Z}(I, J)$ , two filtrations are used to determine the sparse pattern for  $X_k$  as:

$$\begin{aligned} J &= \{j | j \in S \quad \text{dist}(G, j) \leq \tau_1\} \\ I &= \{i | i \in S \quad \text{dist}(i, J) \leq \tau_2\} \end{aligned}, \quad (7)$$

where  $S$  is the sparse pattern of  $Z_{near}$ ,  $\text{dist}(G, j)$  is the distance between the center of box  $G$  containing  $X_k$  and that of the  $j$ th basis function, and  $\text{dist}(i, J)$  is the distance between the center of the  $i$ th testing function and that of boxes containing all basis functions whose index  $j \in J$ . Thus, in order to select  $\bar{Z}(I, J)$  for  $X_k$ , we need to draw a sphere of radius  $\tau_1$  with the same center of box  $G$ . The  $j$ th column entries  $\bar{Z}(:, j)$  will be kept in  $\bar{Z}(:, J)$  if the  $j$ th basis function is exactly contained in this sphere, otherwise  $\bar{Z}(:, j)$  will be dropped. The similar procedure is also executed in the second filtration for row indices selection with a sphere of radius  $\tau_2$ . However, during the first filtration, if the testing function  $X_k$  and two basis functions  $X_{j_1}, X_{j_2}$  reside in the MLFMA leaf boxes (black line) as shown in Fig. 1, then the distance between  $X_k$  and  $X_{j_1}$  will be smaller than  $\tau_1$ , which leads to a strong mutual interaction  $Z_{kj_1}$ . But according to the selection criterion (7),  $Z_{kj_1}$  is dropped. In contrast, although the distance between  $X_k$  and  $X_{j_2}$  is larger than  $\tau_1$ , the resultant matrix entry  $Z_{kj_2}$  that represents a relatively weak interaction between  $X_k$  and  $X_{j_2}$ , is still included in  $\bar{Z}(I, J)$ . The similar situation will occur once again during the second filtration. Therefore, this two-parameter filtration scheme may not utilize the nonzero entries reasonably and effectively in some situations.

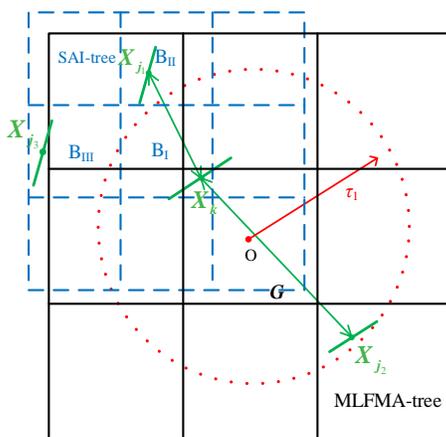


Fig. 1. Locations of  $X_k, X_{j_1}, X_{j_2}$  and  $X_{j_3}$  in the MLFMA leaf boxes and in the SAI-tree.

To solve this problem, we build a new octree structure with smaller box size  $R_{SAI}$  compared with the MLFMA leaf box, called SAI-tree which is shown in Fig. 1 (blue dashed line), to construct the SAIP. That is to say, we can use the SAI-tree to control the sparse pattern and to generate the matrix  $\bar{Z}$ , i.e.,  $\bar{Z}$  is considered as the near-field matrix in the SAI-tree. As shown in Fig. 1, in the SAI-tree, box  $B_{II}$  that contains  $X_{j_1}$  is one of the neighbor boxes of box  $B_I$  containing  $X_k$ , so the entry  $Z_{kj_1}$  representing strong interaction between  $X_k$  and  $X_{j_1}$  will be included in  $\bar{Z}$ . On the contrary, since  $X_{j_2}$  does not belong to any neighbor box of  $B_I$ ,  $Z_{kj_2}$  that denotes the relatively weak interaction between  $X_k$  and  $X_{j_2}$  will be naturally filtered. On the other hand, however, in some extreme situations as shown in Fig. 1, in the SAI-tree the box  $B_{III}$  that contains  $X_{j_3}$  is one of the neighbor boxes of the box  $B_I$  containing  $X_k$ , while in the original octree of the MLFMA, the entry  $Z_{kj_3}$  representing the interaction between  $X_k$  and  $X_{j_3}$  is not included in the near-field matrix  $Z_{near}$ . In this situation,  $Z_{kj_3}$  needs to be calculated individually during generating  $\bar{Z}$ .

The proposed sparse pattern selection scheme exhibits two advantages over the conventional ones:

(1) Effective utilization of the nonzero entries. The proposed scheme can more “accurately” keep the generally strong interactions and drop the weak ones based on the geometrical information provided by the SAI-tree.

(2) Easy implementation. It has only one tuning parameter  $R_{SAI}$ , while  $R_{SAI}$  can be chosen to be smaller than the size of the leaf box in the MLFMA. If we still use  $N_g$  and  $n$  to denote the number of non-empty boxes in the SAI-tree and the average number of nonzero entries in each row of  $\bar{Z}$  respectively, a smaller  $R_{SAI}$  leads to a larger  $N_g$  and a smaller  $n$ , which will further decrease the SAIP setup time since the computational complexity is about  $O(N_g n^3)$  as mentioned previously. Furthermore, the value of  $R_{SAI}$  can be automatically fixed according to the average discretized mesh size, which will greatly simplify the process of tuning an appropriate parameter for a certain problem. It is worth mentioning that the proposed scheme needs additional time to construct the SAI-tree, but due to the utilization of the fast binary-tree searching algorithm, the additional time is very limited.

Applying the proposed sparse pattern selection scheme based on the SAI-tree, the implementation process of constructing  $M$  is described as follows:

(1) Determine the value of  $R_{SAI}$  according to the average discretized mesh size. According to a large amount of numerical experiments,  $R_{SAI}$  is recommended to be two times of the average mesh size.

(2) Build the SAI-tree in which the box size is set to  $R_{SAI}$ .

(3) Select the sparse pattern according to the index of near-field interactions in the SAI-tree.

(4) Generate  $\bar{Z}$  according to the selected sparse pattern. If the entry in  $\bar{Z}$  also belongs to  $Z_{near}$ , then extract it from  $Z_{near}$  directly; if not, calculate it individually.

(5) Construct the preconditioner  $M$  according to (2).

Moreover, for the objects with extremely large electrical size,  $Z_{near}$  becomes insufficient to approximate the entire impedance matrix, due to the inefficiency of the preconditioners constructed from  $Z_{near}$ . One way to alleviate this problem is to use the far-field interactions during constructing the preconditioners [7, 8]. In these approaches, a cheaper MLFMA version called approximate MLFMA is used as the preconditioner to iteratively solve a closely related matrix equation. In the practical process, a secondary preconditioner is needed to accelerate the iterative solving process of the closely related matrix equation. The SAIP applying the proposed sparse pattern selection scheme can be adopted as this secondary preconditioner.

### III. NUMERICAL RESULTS

To validate the effectiveness of the proposed SAIP, EM radiation and scattering of composite dielectric-conductor objects are calculated. In our implementation, the RWG [9] and SWG [10] basis functions are used to model the equivalent surface and volume currents, respectively. The restarted GMRES with a restart parameter  $m$  is used as the iterative solver to reach convergence with relative residual error of 0.001 [11-13]. Zero vector is taken as initial approximate solution for all calculations. The leaf box size of the MLFMA is  $0.25\lambda$  ( $\lambda$  is the wavelength in the free space). All computations are in single precision and carried out on a workstation with 2.4 GHz CPU and 384 GB RAM.

#### A. Microstrip patch antenna array

The first example is the EM radiation of an antenna array composed of 64 microstrip patch antennas at the operating frequency of 14.5 GHz as shown in Fig. 2. All antennas are excited with the same magnitude and phase. The relative dielectric constant is  $\epsilon_r=2.2-j0.00198$ . The average mesh size is about  $0.08\lambda$ , and the number of discrete triangles, tetrahedrons and unknowns are 9,375, 21,062 and 63,340, respectively, while the octree of MLFMA has 6 levels. The restart parameter of GMRES is  $m=100$ . To compare with other kinds of preconditioners limited to sequential implementation such as the preconditioners based on incomplete Lower-Upper decomposition (ILU), the whole computation process is executed serially. Table 1 shows the performance details of three schemes to construct SAIP from selecting different sub-matrices ( $SS$ ,  $SS&VV$ , and  $Tot$ ) with various tuning parameter  $R_{SAI}$ . This table contains the memory usage of preconditioner (PC-Mem), the additional time to construct the SAI-tree and generate  $\bar{Z}$  ( $T_{add}$ ), the setup time ( $T_{PC}$ ) to construct preconditioner, the average number of basis functions in the non-empty boxes of the

SAI-tree ( $\Delta$ ), and so on. Besides, *none* means that the computation is executed without any preconditioner. From Table 1, the first finding is that the three different sub-matrix selection schemes need similar number of iterations to reach the target convergence, while the memory usage and setup time for constructing  $M$  by  $SS$  are much less than those by  $SS&VV$  and  $Tot$ . Therefore, for the VSIE solution, using sub-matrix  $Z_{SS}$  alone to construct the SAIP might be an appropriate choice.

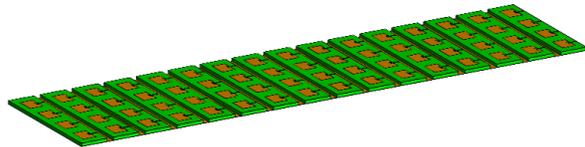


Fig. 2. The structure of antenna array composed of 64 microstrip patch antennas at the operating frequency of 14.5 GHz.

The second finding is that large value of  $R_{SAI}$  does not always mean less total computation time, while the proper value of  $R_{SAI}$  is found to be about twice as large as the average mesh size. The reason is that enlarging the value of  $R_{SAI}$  will make  $\bar{Z}$  denser, which would dramatically increase the setup time and the memory requirement for constructing preconditioner  $M$ . This phenomenon also demonstrates that enlarging the number of nonzero entries in matrix  $\bar{Z}$  does not always lead to better preconditioning performance. It is worth to point out that some effective preconditioners can be constructed based on the Schur complement, which are specially designed to work in conjunction with the VSIE [14]. However, this kind of preconditioners usually needs to calculate both of the inverse of  $Z_{SS}$  and  $Z_{VV}$  directly or approximately. Thus, it is concluded that the memory usage and setup time of this kind of preconditioners based on the Schur complement are on the same level as SAIP by  $SS&VV$ , which are evidently larger than  $SS$ . For this reason, this kind of preconditioners does not be adopted in this paper.

From the above, we may draw the conclusion that in the VSIE solution, using  $Z_{SS}$  alone to construct the SAIP is an appropriate choice. However, whether this  $SS$  scheme can be successfully utilized to other kinds of preconditioners is worthwhile of further research. Table 2 lists the detailed computation information with various kinds of preconditioners on the first example, such as the SAIP with two tuning parameters  $\tau_1$ ,  $\tau_2$  reported in [3] (denoted by *conv*), the ILUT preconditioner [15] which is based on a dual dropping rule with two tuning parameters: the fill-in  $p$  and the threshold drop tolerance  $\tau$ , and the variant approach of ILUT, called SuperLU [16], with three tuning parameters: the fill-ratio  $\gamma$ , the threshold drop tolerance  $\tau$ , and the pivoting threshold  $\eta$ . Besides, only  $Z_{SS}$  is used to construct the preconditioner.

For *conv*, the parameter setting refers to [3]. Since the leaf box size of the MLFMA is  $0.25\lambda$ , then,

$$0.125\lambda \leq \tau_1, \tau_2 \leq 0.5\lambda \quad \text{and} \quad \tau_1 \leq \tau_2. \quad (8)$$

For the ILU type preconditioners, as stated in [15, 16], a more effective and robust ILUT or SuperLU preconditioner can be constructed with larger  $p$  or  $\gamma$ . In fact, when the preconditioning matrix is denser, it is indeed necessary to set a relatively larger  $p$  or  $\gamma$  in the ILUT factorization to achieve better performance. For this calculated antenna array, since it contains thin structures, dense meshes are needed to model the details, due to a relatively dense  $Z_{near}$  as well as  $Z_{SS}$ . As a consequence,  $p$  or  $\gamma$  needs to be set to a large value. From Table 2, it is obtained that when  $p < 200$  or  $\gamma < 10$ , the ILUT or SuperLU preconditioner will be failure. Another finding is that the value of  $\tau$  does not affect very much

the convergence rate during the iterative solution, which is in accordance with [15]. Comparing Table 1 with Table 2, it is found that different kinds of preconditioners improve the convergence in varying degrees. Comparing the proposed scheme (*prop*, i.e., SAIP with *SS* scheme) with *conv*, when an appropriate combination of tuning parameters is used ( $R_{SAIP}=0.15\lambda$  versus  $\tau_1, \tau_2=0.25, 0.5$ ), *prop* shows less iterations (169 versus 384) and less total time (188s versus 323s) to achieve convergence with similar memory usage (1141.8 MB versus 1139.4 MB). Compared with ILUT and SuperLU, *prop* shows the similar robustness, while the memory usage of SAIP is several times less than that of ILUT or SuperLU. More importantly, *prop* can be naturally parallelized, while the ILU-type preconditioners are limited to sequential implementation.

Table 1: Detailed performance of different sub-matrix selection schemes and  $R_{SAIP}$  on the proposed SAIP

| SAIP Style       | $R_{SAIP}$ ( $\lambda$ ) | PC-Mem (MB) | $T_{add}$ (s) | $T_{PC}$ (s) | $n$    | $N_g$ | $\Delta$ | Solution Time (s) | Iterations | Total Time (s) | Peak Memory (MB) |
|------------------|--------------------------|-------------|---------------|--------------|--------|-------|----------|-------------------|------------|----------------|------------------|
| <i>None</i>      | —                        |             |               |              |        |       |          | 549               | 897        | 592            | 1130.6           |
| <i>SS</i>        | 0.125                    | 8.26        | 0.72          | 5.97         | 82.2   | 2228  | 5.9      | 521               | 757        | 583            | 1138.4           |
|                  | 0.15                     | 11.70       | 1.54          | 19.2         | 116.4  | 1479  | 8.9      | 125               | 169        | 188            | 1141.8           |
|                  | 0.2                      | 19.74       | 1.73          | 29.8         | 196.7  | 987   | 13.3     | 116               | 151        | 190            | 1198.3           |
|                  | 0.25                     | 29.54       | 0.17          | 81.5         | 294.3  | 732   | 18.0     | 101               | 129        | 232            | 1209.4           |
| <i>SS&amp;VV</i> | 0.125                    | 113.8       | 0.73          | 306          | 252.5  | 2939  | 21.6     | 556               | 753        | 914            | 1294.9           |
|                  | 0.15                     | 174.3       | 3.28          | 811          | 361.0  | 2048  | 30.9     | 134               | 163        | 987            | 1356.7           |
|                  | 0.2                      | 299.5       | 4.21          | 2421         | 619.5  | 1152  | 55.0     | 132               | 149        | 2599           | 1478.1           |
|                  | 0.25                     | 447.7       | 0.23          | 5342         | 926.4  | 785   | 80.7     | 119               | 127        | 5501           | 1627.2           |
| <i>Tot</i>       | 0.125                    | 183.3       | 0.74          | 816          | 379.3  | 2939  | 21.6     | 581               | 739        | 1442           | 1364.3           |
|                  | 0.15                     | 261.5       | 3.92          | 1641         | 541.1  | 2048  | 30.9     | 136               | 161        | 1823           | 1443.9           |
|                  | 0.2                      | 447.9       | 4.56          | 4713         | 923.5  | 1152  | 55.0     | 137               | 144        | 4892           | 1629.1           |
|                  | 0.25                     | 699.5       | 0.24          | 10852        | 1385.5 | 785   | 80.7     | 124               | 121        | 11021          | 1881.3           |

Table 2: Detailed performance of different types of preconditioners

| Preconditioner Style                | Parameters      | PC-Mem (MB) | $T_{PC}$ (s) | Solution Time (s) | Iterations | Total Time (s) | Peak Memory (MB) |
|-------------------------------------|-----------------|-------------|--------------|-------------------|------------|----------------|------------------|
| <i>Conv</i><br>( $\tau_1, \tau_2$ ) | 0.2, 0.4        | 6.94        | 5.01         | *                 |            |                | 1137.7           |
|                                     | 0.2, 0.5        | 7.92        | 8.66         | 358               | 512        | 402            | 1138.6           |
|                                     | 0.25, 0.5       | 8.79        | 9.33         | 269               | 384        | 323            | 1139.4           |
|                                     | 0.3, 0.5        | 19.8        | 19.6         | 262               | 355        | 325            | 1150.3           |
| ILUT<br>( $p, \tau$ )               | 150, 0.001      | 38.6        | 8.2          | *                 |            |                | 1213.9           |
|                                     | 200, 0.001      | 42.7        | 10.4         | 146               | 178        | 201            | 1218.1           |
|                                     | 200, 0.0001     | 62.8        | 30.3         | 133               | 165        | 206            | 1237.6           |
|                                     | 300, 0.001      | 46.6        | 11.9         | 139               | 172        | 194            | 1220.2           |
| SuperLU<br>( $\gamma, \tau, \eta$ ) | 5, 0.001, 0.5   | 76.1        | 24.3         | *                 |            |                | 1223.1           |
|                                     | 10, 0.001, 0.5  | 108.8       | 32.2         | 138               | 140        | 216            | 1254.0           |
|                                     | 10, 0.0001, 1.0 | 149.0       | 42.9         | 140               | 135        | 239            | 1293.5           |
|                                     | 15, 0.001, 0.5  | 113.2       | 38.1         | 125               | 127        | 208            | 1259.8           |

Note: “\*” refers to no convergence after 1000 iterations.

As mentioned before in this paper, the restarted GMRES, which is a famous Krylov subspace method for

solving nonsymmetric linear systems, is used as the iterative solver. However, as argued in [11-13], the

restart parameter  $m$  will deeply influence the convergence of this Krylov subspace method. In order to investigate how  $m$  influences the performance, Fig. 3 shows the iterations for four preconditioners ( $prop$ ,  $conv$ , ILUT, SuperLU) with respect to different values of  $m$ , while the relative residual error is fixed to 0.001. It is observed that when  $m$  is too small, none of the four preconditioners can prompt GMRES to reach the target convergence after 1000 iterations. Along with the increase of  $m$ , the iterations will sharply decrease, and be followed by a steady decrease. When  $m$  is about 30,  $prop$  can achieve the convergence after hundreds of iterations, while for other three preconditioners,  $conv$ , ILUT and SuperLU can achieve the same convergence when  $m$  is about 60, 50 and 40, respectively. This illustrates that compared with  $conv$ , ILUT and SuperLU,  $prop$  is cooperated with a relatively small  $m$ , which has a low memory usage during using the GMRES to solve the matrix equation. Another finding is that when the value of  $m$  is suitably fixed, the convergence rate of  $prop$  is on the same level as that of the ILU type of preconditioners, while the  $conv$  is relatively slow. Therefore, Fig. 3 illustrates the robustness and efficiency of the proposed approach.

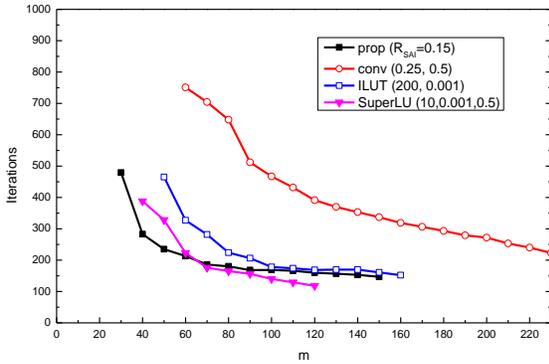


Fig. 3. Number of iterations for different preconditioners with respect to different restart parameter  $m$  in the GMRES.

## B. Dielectric coated conducting sphere

The second example is the calculation of bistatic radar cross section (RCS) of a dielectric coated conducting sphere. The radius of the conducting sphere is  $10\lambda$ , and the thickness of the coating dielectric is  $0.05\lambda$ . The average mesh size is about  $0.085\lambda$  and the total number of triangles, tetrahedrons and unknowns are 168,650, 934,261 and 2,447,096, respectively.  $R_{SAI}$  is set to  $0.17\lambda$  which is about twice as large as the average mesh size. In this calculation, the OpenMP parallel technology is adopted and 16 cores are involved [17]. To verify the performance of SAIP, combined field integral equation (CFIE) or EFIE is adopted to disperse the conducting sphere part, which can be combined with VIE to yield the CFIE-VIE or EFIE-VIE type of VSIE. Since the matrix equation generated by EFIE-VIE is very

ill-conditioned, the restart parameter of GMRES is set to  $m=200$  for EFIE-VIE which is a relatively large value, while  $m=30$  when CFIE-VIE is adopted. Figures 4 and 5 show the number of iterations and solution time required by the SAIP with the three selection schemes ( $SS$ ,  $SS&VV$  and  $Tot$ ) to achieve convergence when the relative dielectric constant  $\epsilon_r$  varies from 1.5 to 9.0, respectively. Besides, when EFIE-VIE is adopted without any preconditioner ( $none$ ),  $\epsilon_r=9$  cannot achieve convergence after 5000 iterations. From Fig. 4 and Fig. 5, it is clear that the proposed SAIP can accelerate the convergence rate obviously, while  $SS$ ,  $SS&VV$  and  $Tot$  need comparable iterations and solution time to reach the target convergence. Table 3 lists the computational details in terms of the memory usage of preconditioner (PC-Mem), the additional time to construct the SAI-tree and generate  $\bar{Z}$  ( $T_{add}$ ), the SAIP setup time ( $T_{PC}$ ), and the peak memory usage with  $m=200$  for  $none$ ,  $SS$ ,  $SS&VV$  and  $Tot$ . It is found that the memory usage and setup time for SAIP by  $SS&VV$  or  $Tot$  are several times as those by  $SS$ , which illustrates the efficiency of  $SS$ . When  $\epsilon_r=2$ , the numerical results are shown in Fig. 6, while the exact result from Mie series is also given as a reference. It is observed that these numerical results have very high calculating precision.

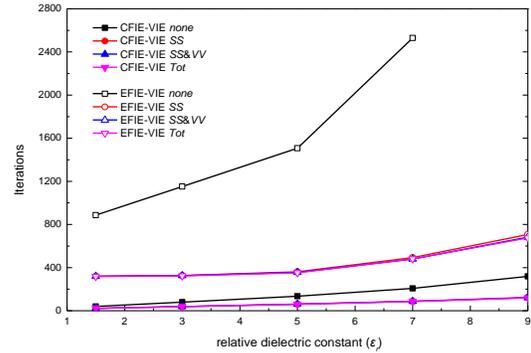


Fig. 4. Number of iterations for various  $\epsilon_r$ , sub-matrix selection schemes, and VSIE formulations.

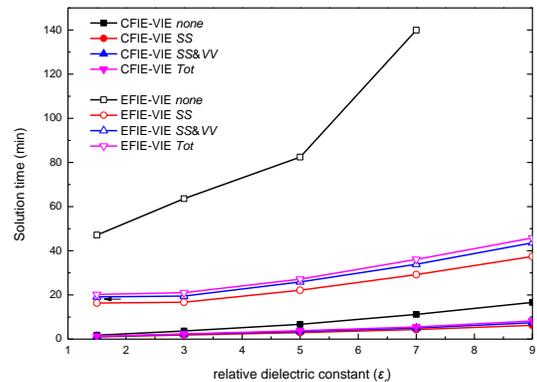


Fig. 5. Solution time for various  $\epsilon_r$ , sub-matrix selection schemes, and VSIE formulations.

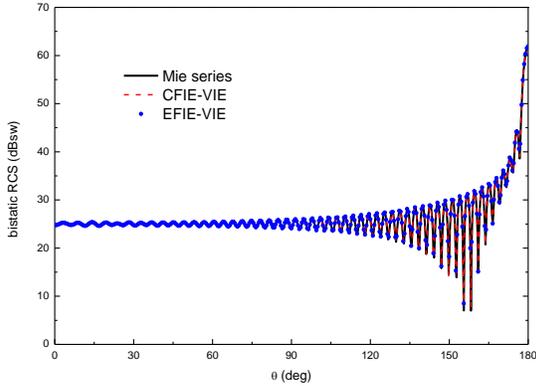


Fig. 6. Bistatic RCS of a conducting sphere of radius  $10\lambda$  coated with  $0.05\lambda$  thick homogeneous dielectric ( $\epsilon_r=2$ ) at  $\varphi=0$  plane, illuminated by a  $\theta$ -polarized plane wave with the incident angle  $\theta^i=0$ ,  $\varphi^i=0$ .

Table 3: Details of different selection schemes from the proposed SAIP with  $R_{SAI}=0.17\lambda$  and  $m=200$  for a conducting sphere of radius  $10\lambda$  coated with  $0.05\lambda$  thick homogeneous dielectric

| SAIP Scheme      | PC-Mem (MB) | $T_{add}$ (s) | $T_{PC}$ (s) | Peak Memory (GB) |
|------------------|-------------|---------------|--------------|------------------|
| <i>None</i>      | —           | —             | —            | 70.2             |
| <i>SS</i>        | 210.1       | 1.3           | 11.8         | 70.4             |
| <i>SS&amp;VV</i> | 15808.7     | 19.8          | 9986.6       | 86.0             |
| <i>Tot</i>       | 19387.2     | 23.2          | 14086.8      | 89.5             |

### C. Dielectric coated conducting almond

The monostatic RCS of a coated PEC almond is calculated, the geometric equation of which is listed in [18], while its position in the Cartesian coordinate is shown in Fig. 7. The length of the PEC almond is  $10\lambda$ , the coating thickness is  $0.05\lambda$ , and the relative dielectric constant of the coated dielectric is  $\epsilon_r=3.38-j0.0338$ . After discretization, the average mesh size is about  $0.075\lambda$  and the total number of triangles, tetrahedrons and unknowns are 30,880, 93,085 and 263,508, respectively. The coated almond is illuminated by a  $\theta$ - or  $\varphi$ -polarization plane wave, and the observation range is  $\theta=90^\circ$  and  $0\leq\varphi\leq90^\circ$  with 91 observation points. During the calculation,  $R_{SAI}$  is set to  $0.15\lambda$ , the OpenMP parallelization with 16 cores is applied, and CFIE-VIE or EFIE-VIE is adopted to disperse the object, while the restart parameter of GMRES is set to  $m=100$  for the EFIE-VIE or  $m=30$  for the CFIE-VIE. Table 4 shows the computational details in terms of the memory usage of SAIP (PC-Mem), the SAIP setup time ( $T_{PC}$ ), and the peak memory usage with  $m=100$  for the calculation process without any preconditioner (*none*) or with

SAIP by different sub-matrix selection schemes (*SS*, *SS&VV* and *Tot*), while Table 5 shows the total time of the whole process in terms of different incident wave polarization modes and different VSIE types. From Table 4, it is found that except *none*, *SS* shows the least peak memory usage and setup time, while Table 5 shows that *SS* leads to the minimum total time for both VSIE types and both polarization modes. Figure 8 shows the number of iterations with respect to different observation angles for different polarizations and VSIE types, respectively. It is observed that compared with *none*, the SAIP with each selection scheme (*SS*, *SS&VV* or *Tot*) can significantly accelerate the convergence. In addition, in terms of the acceleration performance, *SS* is on the same level as *SS&VV* or *Tot*. However, *SS* needs the least setup time and memory usage. This example illustrates that even for the calculation of monostatic RCS whose solution time occupies most of the total calculation time, *SS* also shows competitive effect.

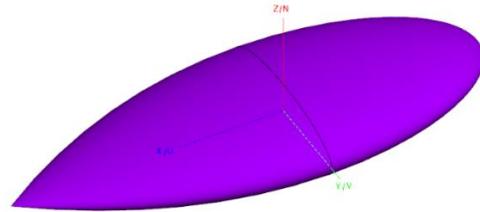


Fig. 7. The coated PEC almond in the Cartesian coordinate.

Table 4: Details of different selection schemes from the proposed SAIP with  $R_{SAI}=0.15\lambda$  and  $m=100$  for a conducting  $10\lambda$  length almond coated with  $0.05\lambda$  thick homogeneous dielectric

| SAIP Scheme      | PC-Mem (MB) | $T_{PC}$ (s) | Peak Memory (GB) |
|------------------|-------------|--------------|------------------|
| <i>None</i>      | —           | —            | 8.83             |
| <i>SS</i>        | 58.1        | 7.1          | 8.89             |
| <i>SS&amp;VV</i> | 1330.9      | 642.5        | 10.1             |
| <i>Tot</i>       | 1874.4      | 1802.3       | 10.6             |

Table 5: The total time of the whole calculation process for a conducting  $10\lambda$  length almond coated with  $0.05\lambda$  thick homogeneous dielectric (unit: minutes)

| VSIE Type        | CFIE-VIE |           | EFIE-VIE |           |
|------------------|----------|-----------|----------|-----------|
|                  | $\theta$ | $\varphi$ | $\theta$ | $\varphi$ |
| <i>None</i>      | 39.7     | 36.0      | 885.5    | 372.3     |
| <i>SS</i>        | 17.2     | 14.1      | 125.1    | 56.8      |
| <i>SS&amp;VV</i> | 27.9     | 23.9      | 134.8    | 65.1      |
| <i>Tot</i>       | 48.2     | 46.0      | 151.5    | 82.7      |

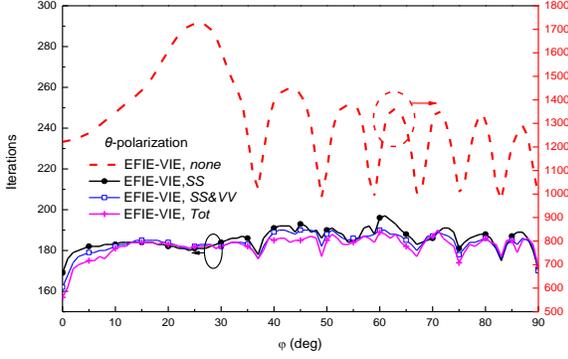
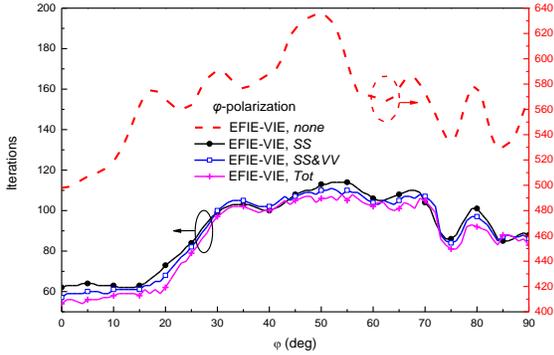
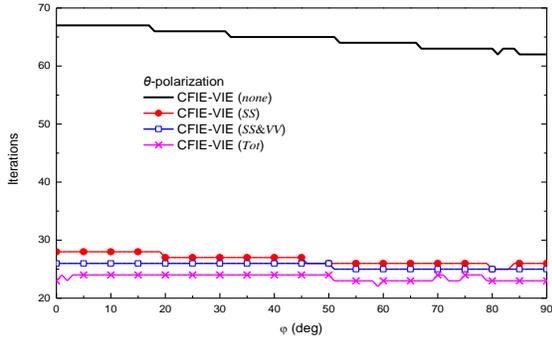
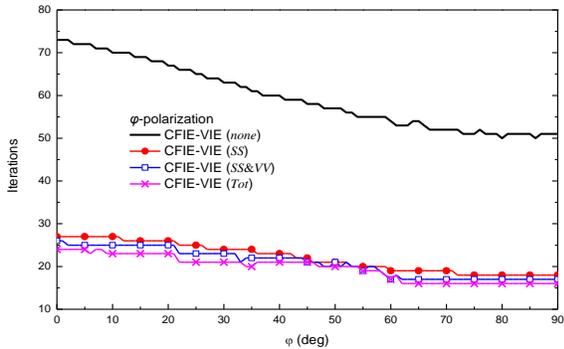
(a)  $\theta$ -polarization incident wave and EFIE-VIE type(b)  $\phi$ -polarization incident wave and EFIE-VIE type(c)  $\theta$ -polarization incident wave and CFIE-VIE type(d)  $\phi$ -polarization incident wave and CFIE-VIE type

Fig. 8. Number of iterations as a function of incident angles for different polarizations and VSIE types.

## IV. CONCLUSIONS

The sub-matrix selection schemes to construct the SAIP for the VSIE solution are discussed. For the VSIE, a high-quality SAIP can be constructed by utilizing the sub-matrix of surface-to-surface interactions in the near-field matrix alone, which significantly reduces the memory usage and setup time. Besides, an effective sparse pattern selection scheme with only one tuning parameter based on the geometrical information provided by the introduced SAI-tree is proposed. Numerical results show that compared with the reported SAIP and ILU-type preconditioners, the proposed SAIP reduces the number of iterations and total computation time with enhanced flexibility and stability, while the one tuning parameter can be automatically set according to the average mesh size.

## ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (61701447, 61671415, and 61331002), and the Fundamental Research Funds for the Central Universities (CUC2019B068).

## REFERENCES

- [1] W. C. Chew, J. M. Jin, E. Michielssen, and J. M. Song, *Fast and Efficient Algorithms in Computational Electromagnetics*, Boston, MA: Artech House, 2001.
- [2] R. F. Harrington, *Field Computation by Moment Methods*, New York: MacMillan, 1968.
- [3] P. L. Rui and R. S. Chen, "An efficient sparse approximate inverse preconditioning for FMM implementation," *Microwave Opt. Technol. Lett.*, vol. 49, No. 7, pp. 1746-1750, July 2007.
- [4] B. Carpentieri, I. S. Duff, L. Giraud, and G. Sylvand, "Combining fast multipole techniques and an approximate inverse preconditioner for large electromagnetism calculations," *SIAM J. Sci. Comput.*, vol. 27, no. 3, pp. 774-792, 2005.
- [5] B. Carpentieri, I. S. Duff, and L. Giraud, "Sparse pattern selection strategies for robust Frobenius-norm minimization preconditioners in electromagnetism," *Numer. Linear Algebra Appl.*, vol. 7, pp. 667-685, 2000.
- [6] J. Lee, J. Zhang, and C. C. Lu, "Sparse inverse preconditioning of multilevel fast multipole algorithm for hybrid integral equations in electromagnetics," *IEEE Trans. Antennas Propag.*, vol. 52, no. 9, pp. 2277-2287, Sept. 2004.
- [7] T. Malas, O. Ergul, and L. Gurel, "Sequential and parallel preconditioners for large-scale integral-equation problems," *2007 IEEE Computational Electromagnetics Workshop*, pp. 35-43, Aug. 2007.
- [8] O. Ergul, T. Malas, and L. Gurel, "Solutions of large-scale electromagnetics problems using an

- iterative inner-outer scheme with ordinary and approximate multilevel fast multipole algorithms,” *Progress in Electromagnetics Research*, vol. 106, pp. 203-223, 2010.
- [9] S. M. Rao, D. R. Wilton, and A. W. Glisson, “Electromagnetic scattering by surfaces of arbitrary shape,” *IEEE Trans. Antennas Propag.*, vol. 30, no. 5, pp. 409-418, May 1982.
- [10] D. H. Schaubert, D. R. Wilton, and A. W. Glisson, “A tetrahedral modeling method for electromagnetic scattering by arbitrarily shaped inhomogeneous dielectric bodies,” *IEEE Trans. Antennas Propag.*, vol. 32, no. 1, pp. 77-85, Jan. 1984.
- [11] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., Philadelphia: SIAM, 2003.
- [12] W. Joubert, “On the convergence behavior of the restarted GMRES algorithm for solving non-symmetric linear systems,” *Numerical Linear Algebra with Applications*, vol. 1, no. 5, pp. 427-447, 1994.
- [13] A. H. Baker, E. R. Jessup, and Tz. V. Kolev, “A simple strategy for varying the restart parameter in GMRES(m),” *Journal of Computational and Applied Mathematics*, vol. 230, no. 2, pp. 751-761, 2009.
- [14] S. Tao, “Block matrix preconditioner for the coupled volume-surface integral equation,” *ACES Journal*, vol. 29, no. 8, pp. 602-610, Aug. 2014.
- [15] J. Lee, J. Zhang, and C. C. Lu, “Incomplete LU preconditioning for large scale dense complex linear systems from electromagnetic wave scattering problems,” *Journal of Computational Physics*, vol. 185, pp. 158-175, Feb. 2003.
- [16] X. S. Li and M. Shao, “A supernodal approach to incomplete LU factorization with partial pivoting,” *ACM Trans. Math. Software*, vol. 37, no. 4, Article no. 43, Apr. 2011.
- [17] J. Liu, M. He, K. Zhang, B. Wang, and Q. Qiu, “Parallelization of the multilevel fast multipole algorithm by combined use of OpenMP and VALU hardware acceleration,” *IEEE Trans. Antennas Propag.*, vol. 62, no. 7, pp. 3884-3889, July 2014.
- [18] A. C. Woo, H. T. G. Wang, M. J. Schuh, and M. L. Sanders, “EM programmer's notebook-benchmark radar targets for the validation of computational electromagnetics programs,” *IEEE Antennas Propag. Magazine*, vol. 35, no. 1, pp. 84-89, Feb. 1993.