Accuracy Controlled Structure-Preserving $\mathcal{H}^2$-Matrix-Matrix Product in Linear Complexity With Change of Cluster Bases

Miaomiao Ma, *Graduate Student Member, IEEE*, and Dan Jiao, *Fellow, IEEE*

***Abstract*— $\mathcal{H}^2$-matrix constitutes a general mathematical framework for efficient computation of both partial-differential-equation (PDE) and integral-equation (IE)-based operators. Existing linear-complexity $\mathcal{H}^2$ matrix-matrix product (MMP) algorithm lacks explicit accuracy control, while controlling accuracy without compromising linear complexity is challenging. In this article, we develop an accuracy controlled $\mathcal{H}^2$ MMP algorithm by instantaneously changing the cluster bases during the matrix product computation based on prescribed accuracy. Meanwhile, we retain the computational complexity of the overall algorithm to be linear. Different from the existing $\mathcal{H}^2$ MMP algorithm where formatted multiplications are performed using the original cluster bases, in the proposed algorithm, all additions and multiplications are either exact or computed based on prescribed accuracy. Furthermore, the original $\mathcal{H}^2$-matrix structure is preserved in the matrix product. While achieving optimal complexity for constant-rank matrices, the computational complexity of the proposed algorithm is also minimized for variable-rank $\mathcal{H}^2$-matrices. For example, it has a complexity of $O(NlogN)$ for computing electrically large volume IEs, where N is matrix size. The proposed work serves as a fundamental arithmetic in the development of fast solvers for large-scale electromagnetic analysis. Applications to both large-scale capacitance extraction and electromagnetic scattering problems involving millions of unknowns on a single core have demonstrated the accuracy and efficiency of the proposed algorithm.**

***Index Terms*— $\mathcal{H}^2$-matrix, controlled accuracy, electromagnetic analysis, linear complexity, matrix-matrix product (MMP).**

I. Introduction

THE $\mathcal{H}^2$-matrix [1], [2] constitutes a general mathematical framework for compact representation and efficient computation of large dense systems. Both partial differential equation (PDE) and integral equation (IE) operators in electromagnetics can be represented as $\mathcal{H}^2$-matrices with controlled accuracy [3]–[6].

The development of $\mathcal{H}^2$-matrix arithmetic such as addition, multiplication, and inverse are of critical importance to the development of fast solvers in electromagnetics [7]. Take the matrix-matrix product (MMP) as an example, it can be used to efficiently compute the field solution for multiple right-hand sides [8] by representing the right-hand side matrix as an $\mathcal{H}^2$-matrix, and computing the product of the $\mathcal{H}^2$-matrix representation of the original system matrix's inverse and the right-hand side matrix. It can also be used to compute the intermediate matrix such as a Schur complement required in a fast solver [9]. Under the $\mathcal{H}^2$-matrix framework, it has been shown that an $\mathcal{H}^2$-matrix-based addition, matrix-vector product (MVP), and MMP all can be performed in linear complexity for constant-rank $\mathcal{H}^2$ [1]. However, the accuracy of existing $\mathcal{H}^2$-MMP algorithms like [1] is not controlled. This is because given two $\mathcal{H}^2$-matrices $\mathbf{A}_{\mathcal{H}^2}$ and $\mathbf{B}_{\mathcal{H}^2}$, the matrix structure and cluster bases of their product $\mathbf{C} = \mathbf{A}_{\mathcal{H}^2} \times \mathbf{B}_{\mathcal{H}^2}$ are preassumed, and a formatted multiplication is performed, whose accuracy is not controlled. For example, the row cluster bases of $\mathbf{A}_{\mathcal{H}^2}$ and the column cluster bases of $\mathbf{B}_{\mathcal{H}^2}$ are assumed to be those of $\mathbf{C}$. This treatment lacks accuracy control since the original cluster basis may not be able to represent the new matrix content generated during the MMP. For instance, when multiplying a full-matrix block $\mathbf{F}$ by a low rank block $\mathbf{V}_t\mathbf{S}\mathbf{V}_s{}^T$, treating the result as a low-rank block is correct. Nevertheless, it is inaccurate to use the original row cluster basis $\mathbf{V}_t$ as the product's row cluster basis, since the latter has been changed to $\mathbf{F}\mathbf{V}_t$. Therefore, the algorithm in [1] can be accurate if the cluster bases of the original matrices can also be used to accurately represent the matrix product. However, this is unknown in general applications, and hence the accuracy of existing linear-complexity MMP algorithm is not controlled. One can find many cases where a formatted multiplication would fail.

The posteriori multiplication in [2] is more accurate than the formatted multiplication in [1]. But it is only suitable for special $\mathcal{H}^2$ matrices. For example, in this special $\mathcal{H}^2$-matrix, if one inadmissible block is multiplied by another inadmissible block, then the product must be treated as an inadmissible block. In general, when computing $\mathbf{C} = \mathbf{A}_{\mathcal{H}^2} \times \mathbf{B}_{\mathcal{H}^2}$, the block partitions in $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are determined by the admissibility condition. It is common to encounter the case that two inadmissible blocks are multiplied, but the target block in $\mathbf{C}$ is admissible. In addition, this posteriori multiplication requires much more computational time and memory than the formatted one. It needs to first represent the product in an $\mathcal{H}$-matrix and then convert it into an $\mathcal{H}^2$-matrix, the complexity of which

Manuscript received July 31, 2019; revised October 15, 2019 and December 5, 2019; accepted December 15, 2019. Date of publication January 28, 2020; date of current version January 31, 2020. This work was supported by Defense Advanced Research Projects Agency (DARPA) under Award FA8650-18-2-7847. This article is an expanded version from the IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization, Cambridge, MA, USA, May. 29–31, 2019. *(Corresponding author: Dan Jiao.)*

The authors are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: djiao@purdue.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMTT.2020.2967718

is not linear. In [10], local low-rank updates are performed to control the accuracy of MMP. However, the computational cost of each update is $O[k^2(\#t+\#s)]$, which depends on the row dimension ($\#t$) and column dimension ($\#s$) of the matrix blocks. For constant-rank $\mathcal{H}^2$-matrix, i.e., constant k, this leads to a complexity of $O(k^2 N log N)$. Nonetheless, for rank that increases with the electrical size like that encountered in the electrically large analysis, the resultant complexity can become very high. For example, for k that linearly increases with electrical size, and hence being a function of N, the complexity of MMP of [10] would be at least $O(N^{5/3})$ in a volumetric analysis, and even higher in a surface analysis.

In this article, we propose a new algorithm to do the $\mathcal{H}^2$ matrix-matrix multiplication with controlled accuracy. The cluster bases are calculated instantaneously based on the prescribed accuracy during the computation of the MMP. Meanwhile, we are able to keep the computational complexity to be linear for constant-rank $\mathcal{H}^2$. For variable-rank cases such as those in an electrically large analysis, the proposed MMP is also efficient since it only involves $O(2^l)$ computations at level l, each of which costs $O(k_l^3)$ only, where k_l is the rank at tree level l. This algorithm can be used as a fundamental arithmetic in the error-controlled fast inverse, lower-upper (LU) factorization, solution for many right-hand sides, and so on. Numerical experiments have demonstrated its accuracy and low complexity. In [17] and [18], we present a fast algorithm to compute the product of two $\mathcal{H}^2$-matrices in controlled accuracy. However, unlike this article, the original cluster bases are not completely changed, but appended to account for the updates to the original matrix during the MMP. In [12], we present the basic idea of this article. However, it is a one-page abstract. In this article, we provide a complete algorithm together with a comprehensive analysis of its accuracy and complexity, whose validity and performance are then demonstrated by abundant numerical examples.

II. Preliminaries

In an $\mathcal{H}^2$-matrix [1], the entire matrix is partitioned into multilevel admissible and inadmissible blocks, where inadmissible blocks are at the leaf level, noted as $\mathbf{F}_{t,s}$. An admissible matrix block $\mathbf{R}_{t,s}$ satisfies the following strong admissibility condition:

$$\max\{\text{diam}(\Omega_t), \text{diam}(\Omega_s)\} \leq \eta \text{dist}(\Omega_t, \Omega_s) \tag{1}$$

where Ω_t (Ω_s) denotes the geometrical support of the unknown set t (s), diam$\{\cdot\}$ is the Euclidean diameter of a set, dist$\{\cdot,\cdot\}$ denotes the Euclidean distance between two sets, and η is a positive parameter that can be used to control the admissibility condition. An admissible matrix block in an $\mathcal{H}^2$-matrix is represented as

$$\mathbf{R}_{t,s} = (\mathbf{V}_t)_{\#t\times k}(\mathbf{S}_{t,s})_{k\times k}(\mathbf{V}_s)^T_{\#s\times k} \tag{2}$$

where $\mathbf{V}_t$ ($\mathbf{V}_s$) is called cluster basis associated with cluster t (s), $\mathbf{S}_{t,s}$ is called coupling matrix. The cluster bases $\mathbf{V}$ in an $\mathcal{H}^2$-matrix has a nested property. This means the cluster basis for a nonleaf cluster t, $\mathbf{V}_t$, can be expressed by its two children's cluster bases, $\mathbf{V}_{t_1}$ and $\mathbf{V}_{t_2}$, as

$$(\mathbf{V}_t)_{\#t\times k} = \begin{bmatrix} (\mathbf{V}_{t_1})_{\#t_1\times k_1} & 0 \\ 0 & (\mathbf{V}_{t_2})_{\#t_2\times k_2} \end{bmatrix} \begin{bmatrix} (\mathbf{T}_{t_1})_{k_1\times k} \\ (\mathbf{T}_{t_2})_{k_2\times k} \end{bmatrix} \tag{3}$$

where $\mathbf{T}_{t_1}$ and $\mathbf{T}_{t_2}$ are called transfer matrices. Because of such a nested relationship, the cluster bases only need to be stored for leaf clusters. For nonleaf clusters, only transfer matrices need to be stored. The $\mathcal{H}^2$-matrix is stored in a tree structure, with the size of leaf-level clusters denoted by $leafsize$. The number of blocks formed by a single cluster at each tree level is bounded by a constant due to the way that the matrix is partitioned in an $\mathcal{H}^2$-matrix. This constant is denoted by C_{sp}. In an $\mathcal{H}^2$-matrix, a large matrix block consisting of $\mathbf{F}$ and $\mathbf{R}$ is called a nonleaf block $\mathbf{NL}$. As an example, a four-level block cluster $\mathcal{H}^2$-tree is illustrated in Fig. 1(a), where the green link connects a row cluster with a column cluster, which form an admissible block, and the red links are for inadmissible blocks. The resultant $\mathcal{H}^2$-matrix is shown in Fig. 1(b), where the admissible blocks are marked in green and the inadmissible blocks are marked in red.

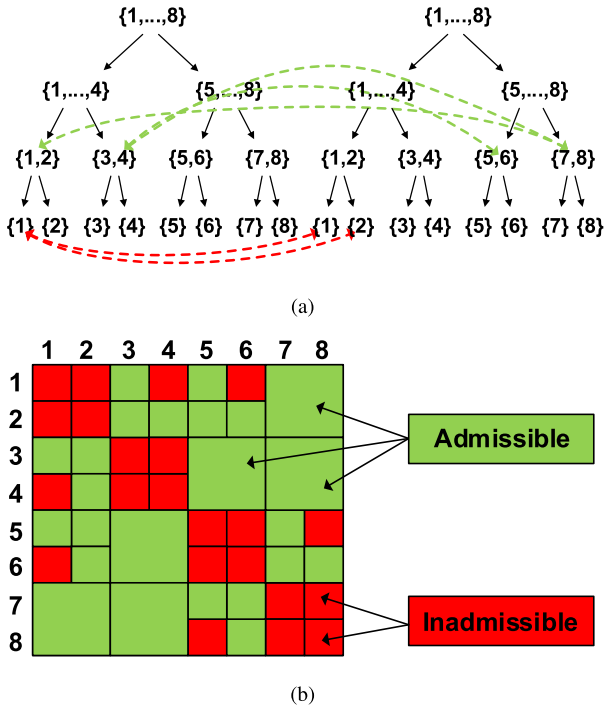


Fig. 1. Illustration of a block cluster tree and resulting $\mathcal{H}^2$-matrix partition. (a) Block cluster tree. (b) $\mathcal{H}^2$-matrix structure.

III. Proposed $\mathcal{H}^2$ MMP Algorithm—Leaf Level

To compute $\mathbf{A}_{\mathcal{H}^2} \times \mathbf{B}_{\mathcal{H}^2} = \mathbf{C}_{\mathcal{H}^2}$, unlike the existing $\mathcal{H}^2$ formatted MMP [1], which is recursive, we propose to perform a one-way tree traversal from leaf level all the way up to the minimum level that has admissible blocks. Here, the tree is inverted with root level at level 0. While doing the multiplications at each level, we instantaneously compute the new row and column cluster bases based on prescribed accuracy to represent the product matrix accurately. We will use the $\mathcal{H}^2$-matrices shown in Fig. 2 to illustrate the proposed algorithm, but the algorithm is valid for any $\mathcal{H}^2$-matrix.

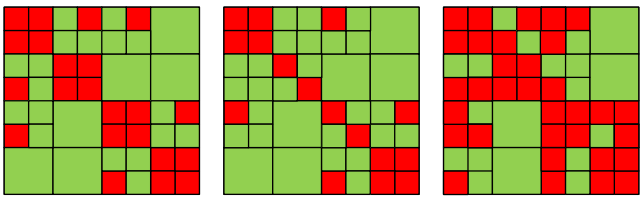


Fig. 2. $\mathcal{H}^2$-matrix structure. (a) $\mathbf{A}_{\mathcal{H}^2}$. (b) $\mathbf{B}_{\mathcal{H}^2}$. (c) $\mathbf{C}_{\mathcal{H}^2}$.

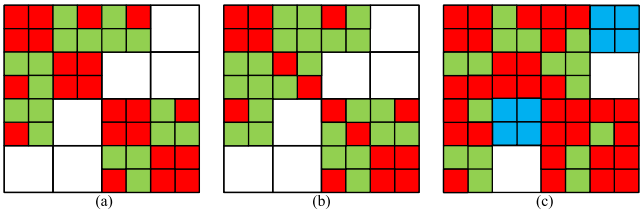


Fig. 3. $\mathcal{H}^2$-matrix at leaf level. (a) $\mathbf{A}^L_{\mathcal{H}^2}$. (b) $\mathbf{B}^L_{\mathcal{H}^2}$. (c) $\mathbf{C}^L_{\mathcal{H}^2}$.

The structures of $\mathbf{A}_{\mathcal{H}^2}$, $\mathbf{B}_{\mathcal{H}^2}$, and $\mathbf{C}_{\mathcal{H}^2}$ matrices, i.e., which block is admissible and which is inadmissible, are determined based on the admissibility condition given in (1). During the product calculation, we will keep the structure of product $\mathbf{C}_{\mathcal{H}^2}$ matrix while achieving prescribed accuracy. In this section, we detail the proposed algorithm for leaf-level multiplications.

We start from leaf level ($l = L$). Let $\mathbf{F}$ denote an inadmissible block, which is stored as a full matrix, and $\mathbf{R}$ be an admissible block. At leaf level, there are in total four matrix-matrix multiplication cases.

1) Case-1: $\mathbf{F}^{\mathbf{A}} \times \mathbf{F}^{\mathbf{B}}$.
2) Case-2: $\mathbf{F}^{\mathbf{A}} \times \mathbf{R}^{\mathbf{B}}$.
3) Case-3: $\mathbf{R}^{\mathbf{A}} \times \mathbf{F}^{\mathbf{B}}$.
4) Case-4: $\mathbf{R}^{\mathbf{A}} \times \mathbf{R}^{\mathbf{B}}$.

The resulting matrix block in $\mathbf{C}$ is of two kinds. First, full matrix block, denoted by $\mathbf{F}^{\mathbf{C}}$, marked in red in Fig. 3(c). Second, admissible block of leaf size, which could be located at leaf level, denoted by $\mathbf{R}^{\mathbf{C},L}$ as marked in green in Fig. 3(c), which could also appear as a subblock in the nonleaf level l as marked in blue in Fig. 3(c). The blue blocks in Fig. 3(c) are only for temporary storage, which will be changed to green admissible blocks during the upper-level multiplication to preserve the structure of $\mathbf{C}_{\mathcal{H}^2}$ matrix. The white blocks in Fig. 3 denote those blocks that are not involved in the leaf-level multiplication. Next, we show how to perform each matrix-matrix multiplication based on the two kinds of target blocks.

A. Product is an Inadmissible Block (Full Matrix) in $\mathbf{C}$

If the product matrix is a full block $\mathbf{F}^{\mathbf{C}}$, we can perform the four cases of multiplications exactly as they are by full matrix multiplications. For the admissible leaf blocks in four cases, we convert them into full matrices and then compute products. Since the size of these matrices is of $leafsize$, a user-defined constant, the computational cost is constant for each of such computations.

B. Product is an Admissible Block in $\mathbf{C}$

If the product is admissible in $\mathbf{C}$ whether it is a leaf-level block or a subblock of a nonleaf admissible block, case-4 can be performed as it is since the product matrix is obviously admissible, which also preserves the original row and column cluster bases. In other words, the row cluster basis of $\mathbf{A}$ is that of $\mathbf{C}$; and the column cluster basis of $\mathbf{B}$ is kept in $\mathbf{C}$. To see this point clearly, we can write

$$\textbf{case-4: } \mathbf{R}^{\mathbf{A}}_{i,j} \times \mathbf{R}^{\mathbf{B}}_{j,k} = \mathbf{V}^{\mathbf{A}}_{i_r} \mathbf{S}^{\mathbf{A}}_{i,j} \left(\mathbf{V}^{\mathbf{A}}_{j_c}\right)^T \times \mathbf{V}^{\mathbf{B}}_{j_r} \mathbf{S}^{\mathbf{B}}_{j,k} \left(\mathbf{V}^{\mathbf{B}}_{k_c}\right)^T \quad (4)$$

where subscripts i, j, and k denote cluster index, subscript r denotes the corresponding cluster is a row cluster, whereas c denotes the cluster is a column cluster. For example, $\mathbf{V}^{\mathbf{A}}_{i_r}$ denotes the cluster basis of row cluster i in $\mathbf{A}$, and $\mathbf{V}^{\mathbf{B}}_{k_c}$ denotes the cluster basis of column cluster k in $\mathbf{B}$. Equation (4) can be written in short as

$$\mathbf{R}^{\mathbf{A}}_{i,j} \times \mathbf{R}^{\mathbf{B}}_{j,k} = \mathbf{V}^{\mathbf{A}}_{i_r} \mathbf{S}^{\mathbf{C}}_{i,k} \left(\mathbf{V}^{\mathbf{B}}_{k_c}\right)^T \quad (5)$$

in which $\mathbf{S}^{\mathbf{C}}_{i,k}$ is the part in between the two cluster bases, which denotes the coupling matrix of the product admissible block in $\mathbf{C}$. Clearly, this case of multiplication does not change the original row and column cluster bases.

For the other three cases, in existing MMP algorithms, a formatted multiplication is performed, which is done in the same way as case-4, i.e., using the original cluster bases of $\mathbf{A}$ and $\mathbf{B}$ or preassumed bases as the cluster bases of the product block. This obviously can be inaccurate since cases-1–3, if performed as they are, would result in different cluster bases in the product matrix, which cannot be assumed. Specifically, case-1 results in a different row as well as column cluster bases in the product admissible block because

$$\textbf{case-1: } \mathbf{F}^{\mathbf{A}}_{i,j} \times \mathbf{F}^{\mathbf{B}}_{j,k} \quad (6)$$

case-2 yields a different row cluster basis since

$$\textbf{case-2: } \mathbf{F}^{\mathbf{A}}_{i,j} \times \mathbf{R}^{\mathbf{B}}_{j,k} = \left(\mathbf{F}^{\mathbf{A}}_{i,j} \mathbf{V}^{\mathbf{B}}_{j_r}\right) \times \mathbf{S}^{\mathbf{B}}_{j,k} \times \left(\mathbf{V}^{\mathbf{B}}_{k_c}\right)^T \quad (7)$$

whereas case-3 results in a different column cluster basis in the product admissible block, because

$$\textbf{case-3: } \mathbf{R}^{\mathbf{A}}_{i,j} \times \mathbf{F}^{\mathbf{B}}_{j,k} = \mathbf{V}^{\mathbf{A}}_{i_r} \times \mathbf{S}^{\mathbf{A}}_{i,j} \times \left(\left(\mathbf{V}^{\mathbf{A}}_{j_c}\right)^T \mathbf{F}^{\mathbf{B}}_{j,k}\right). \quad (8)$$

If we do not update the cluster bases in the product matrix, the accuracy of the multiplication is not controllable. Therefore, in the proposed algorithm, we update row and column cluster bases for multiplication cases 1–3 based on prescribed accuracy. We also have to do so with the nested property taken into consideration so that the computation at nonleaf levels can be performed efficiently.

For case-1, both row and column cluster bases of the product block need to be updated. For case-2, we need to use $\mathbf{F}^{\mathbf{A}}_{i,j} \mathbf{V}^{\mathbf{B}}_{j_r}$ to update the original row cluster basis $\mathbf{V}^{\mathbf{A}}_{i_r}$. For case-3, we need to use $(\mathbf{V}^{\mathbf{A}}_{j_c})^T \mathbf{F}^{\mathbf{B}}_{j,k}$ to update column cluster basis $\mathbf{V}^{\mathbf{B}}_{k_c}$. Since there are many case-1–3 products encountered at the leaf level for the same row or column cluster, we develop the following algorithm to systematically update the cluster bases. In this procedure, we also have to take the computation at all nonleaf levels into consideration so that the changed cluster bases at the leaf level can be reused at the nonleaf levels. To achieve this goal, when we update the cluster basis due to the case-1–3 multiplications associated with this cluster, not only we consider the product admissible block in the leaf level,

but also the admissible blocks at all nonleaf levels. In other words, when computing $\mathbf{A}_{i,j}$ multiplied by $\mathbf{B}_{j,k}$, if the $\mathbf{C}_{i,k}$ block is part of a nonleaf admissible block, we will take the corresponding multiplication into account to update the cluster bases. The detailed algorithms are as follows.

C. Computation of New Cluster Bases in Matrix Product $\boldsymbol{C}_{\mathcal{H}^2}$

First, we show how to calculate the new row cluster bases of $\mathbf{C}_{\mathcal{H}^2}$. Take an arbitrary row cluster i as an example, let its cluster basis in $\mathbf{C}$ be denoted by $\mathbf{V}_{i_r}^{\mathbf{C}}$. This cluster basis is affected by both case-1 and case-2 multiplications, as analyzed earlier. We first find all the case-1 multiplications associated with cluster i, i.e., all $\mathbf{F}_{i,j}^{\mathbf{A}} \times \mathbf{F}_{j,k}^{\mathbf{B}}$ whose product block $\mathbf{C}_{i,k}$ is admissible. Again, notice that the $\mathbf{C}_{i,k}$ can be either admissible at leaf level or be part of a nonleaf admissible block. For any cluster i, the number of $\mathbf{F}_{i,j}^{\mathbf{A}}$ is bounded by constant C_{sp}, since the number of inadmissible blocks that can be formed by a cluster is bounded by C_{sp}. For the same reason, the number of $\mathbf{F}_{j,k}^{\mathbf{B}}$ for cluster j is also bounded by constant C_{sp}. Hence, the total number of $\mathbf{F}_{i,j}^{\mathbf{A}} \times \mathbf{F}_{j,k}^{\mathbf{B}}$ multiplications is bounded by C_{sp}^2, thus also a constant. Then, we calculate the Gram matrix sum of these products as

$$\mathbf{G}_{i_{r1}}^{\mathbf{C},L} = \sum_{j=1}^{O(C_{\text{sp}})} \sum_{k=1}^{O(C_{\text{sp}})} \left(\mathbf{F}_{i,j}^{\mathbf{A}} \mathbf{F}_{j,k}^{\mathbf{B}}\right)\left(\mathbf{F}_{i,j}^{\mathbf{A}} \mathbf{F}_{j,k}^{\mathbf{B}}\right)^H \tag{9}$$

in which superscript H denotes a Hermitian transpose, i.e., a conjugate transpose. We also find all case-2 products associated with cluster i, which is the number of $\mathbf{F}_{i,j}^{\mathbf{A}}$ formed by cluster i at leaf level in $\mathbf{A}_{\mathcal{H}^2}$. This is also bounded by C_{sp}. Since in case-2 products, $\mathbf{F}_{i,j}^{\mathbf{A}}$ is multiplied by an admissible block in $\mathbf{B}$, and hence $\mathbf{V}_{j_r}^{\mathbf{B}}$, we compute

$$\mathbf{G}_{i_{r2}}^{\mathbf{C},L} = \sum_{j=1}^{O\left(C_{\text{sp}}\right)} \left(\mathbf{F}_{i,j}^{\mathbf{A}} \mathbf{V}_{j_r}^{\mathbf{B}}\right)\left(\mathbf{F}_{i,j}^{\mathbf{A}} \mathbf{V}_{j_r}^{\mathbf{B}}\right)^H \tag{10}$$

which incorporates all of the new cluster bases information due to case-2 products.

For case-3 and case-4 multiplications, the row cluster bases of $\mathbf{A}_{\mathcal{H}^2}$ matrix are kept to be those of $\mathbf{C}$. So we account for the contribution of $\mathbf{V}_{i_r}^{\mathbf{A}}$ as

$$\mathbf{G}_{i_{r3}}^{\mathbf{C},L} = \mathbf{V}_{i_r}^{\mathbf{A}} \left(\mathbf{V}_{i_r}^{\mathbf{A}}\right)^H. \tag{11}$$

The column space spanning $\mathbf{G}_{i_{r1}}^{\mathbf{C},L}$, $\mathbf{G}_{i_{r2}}^{\mathbf{C},L}$, and $\mathbf{G}_{i_{r3}}^{\mathbf{C},L}$ would be the new cluster basis of i, since it takes both the original cluster basis and the change to the cluster basis due to matrix products into consideration. Since the magnitude of the three matrices may differ greatly, we normalize them before summing them up so that each component is captured. We thus obtain

$$\mathbf{G}_{i_r}^{\mathbf{C},L} = \widehat{\mathbf{G}_{i_{r1}}^{\mathbf{C},L}} + \widehat{\mathbf{G}_{i_{r2}}^{\mathbf{C},L}} + \widehat{\mathbf{G}_{i_{r3}}^{\mathbf{C},L}}. \tag{12}$$

The $\widehat{\ }$ above $\mathbf{G}_{i_{r1}}^{\mathbf{C},L}$, $\mathbf{G}_{i_{r2}}^{\mathbf{C},L}$, and $\mathbf{G}_{i_{r3}}^{\mathbf{C},L}$ denotes a normalized matrix. There are many ways of normalization. In the examples simulated in this article, the normalization is done by dividing the matrix by its maximum absolute value. We then perform an singular value decomposition (SVD) on $\mathbf{G}_{i_{r3}}^{\mathbf{C},L}$ to obtain the row cluster bases for cluster i of $\mathbf{C}_{\mathcal{H}^2}$ based on prescribed accuracy ϵ_{trunc}. Notice that the cost of SVD at the leaf level is $O(leafsize^3)$, which is a constant. The singular vectors whose normalized singular values are greater than ϵ_{trunc} make the new row cluster basis $\mathbf{V}_{i_r}^{\mathbf{C}}$. It can be used to accurately represent the admissible blocks related to cluster i in $\mathbf{C}_{\mathcal{H}^2}$. Here, notice that the proposed algorithm for computing matrix-product cluster bases keeps nested property of $\mathbf{V}_{i_r}^{\mathbf{C}}$. This is because the Gram matrix sums in (9), (10), and (11) take the upper-level admissible products into account.

To compute the column cluster bases in $\mathbf{C}_{\mathcal{H}^2}$, the steps are similar to the row cluster basis computation. We account for the contributions from all the four cases of products to compute column cluster bases. As can be seen from (6) and (8), in case-1 and case-3 products, the column cluster bases are changed from the original ones; whereas in case-2 and case-4 products, the column cluster bases are kept the same as those in $\mathbf{B}$.

Consider an arbitrary column cluster k in $\mathbf{C}_{\mathcal{H}^2}$. We find all of the case-1 products associated with k, which is $\mathbf{F}_{i,j}^{\mathbf{A}} \times \mathbf{F}_{j,k}^{\mathbf{B}}$ with target $\mathbf{C}_{i,k}$ being admissible either at the leaf or nonleaf level. The number of such multiplications is bounded by C_{sp}^2. We then compute the sum of their Gram matrices as

$$\mathbf{G}_{k_{c1}}^{\mathbf{C},L} = \sum_{i=1}^{O(C_{\text{sp}})} \sum_{j=1}^{O(C_{\text{sp}})} \left(\mathbf{F}_{i,j}^{\mathbf{A}} \mathbf{F}_{j,k}^{\mathbf{B}}\right)^T \left(\mathbf{F}_{i,j}^{\mathbf{A}} \mathbf{F}_{j,k}^{\mathbf{B}}\right)^*. \tag{13}$$

Here, the superscript * denotes a complex conjugate. We also find all of the case-3 products associated with k, which is $\mathbf{R}_{i,j}^{\mathbf{A}} \times \mathbf{F}_{j,k}^{\mathbf{B}}$ with target $\mathbf{C}_{i,k}$ being admissible either at the leaf or nonleaf level. Hence, the new column cluster basis takes a form of $(\mathbf{V}_{j_c}^{\mathbf{A}})^T \times \mathbf{F}_{j,k}^{\mathbf{B}}$. The number of such multiplications is also bounded by C_{sp}. The sum of their Gram matrices can be computed as

$$\mathbf{G}_{k_{c2}}^{\mathbf{C},L} = \sum_{j=1}^{O(C_{\text{sp}})} \left(\left(\mathbf{V}_{j_c}^{\mathbf{A}}\right)^T \mathbf{F}_{j,k}^{\mathbf{B}}\right)^T \left(\left(\mathbf{V}_{j_c}^{\mathbf{A}}\right)^T \mathbf{F}_{j,k}^{\mathbf{B}}\right)^*. \tag{14}$$

For case-2 and case-4 products, the original column cluster bases of $\mathbf{B}_{\mathcal{H}^2}$ are kept in $\mathbf{C}_{\mathcal{H}^2}$, hence, we compute

$$\mathbf{G}_{k_{c3}}^{\mathbf{C},L} = \mathbf{V}_{k_c}^{\mathbf{B}} \left(\mathbf{V}_{k_c}^{\mathbf{B}}\right)^H. \tag{15}$$

We also normalize these three Gram matrices $\mathbf{G}_{k_{c1}}^{\mathbf{C},L}$, $\mathbf{G}_{k_{c2}}^{\mathbf{C},L}$, and $\mathbf{G}_{k_{c3}}^{\mathbf{C},L}$ and sum them up as

$$\mathbf{G}_{k_c}^{\mathbf{C},L} = \widehat{\mathbf{G}_{k_{c1}}^{\mathbf{C},L}} + \widehat{\mathbf{G}_{k_{c2}}^{\mathbf{C},L}} + \widehat{\mathbf{G}_{k_{c3}}^{\mathbf{C},L}}. \tag{16}$$

We then perform an SVD on this $\mathbf{G}_{k_c}^{\mathbf{C},L}$ and truncate the singular values based on prescribed accuracy ϵ_{trunc} to obtain the column cluster bases $\mathbf{V}_{k_c}^{\mathbf{C}}$ for cluster k. Now, this new column cluster basis $\mathbf{V}_{k_c}^{\mathbf{C}}$ can be used to accurately represent the admissible blocks formed by column cluster k in $\mathbf{C}_{\mathcal{H}^2}$. Notice that the cost of SVD at the leaf level is $O(leafsize^3)$, which is a constant.

D. Computation of the Four Cases of Multiplications With the Product Block Being Admissible

After computing the new row and column cluster bases of the product matrix, for the multiplication whose target is an admissible block described in Section III-B, the computation becomes the coupling matrix computation since the cluster bases have been generated. For the four cases of multiplications, their coupling matrices have the following expressions:

$$\mathbf{S}^{\mathbf{C}}_{i,k} = \begin{cases} \left(\mathbf{V}^{\mathbf{C}}_{i_r}\right)^H \mathbf{F}^{\mathbf{A}}_{i,j} \mathbf{F}^{\mathbf{B}}_{j,k} \left(\mathbf{V}^{\mathbf{C}}_{k_c}\right)^*, & \text{case-1} \\ \left(\mathbf{V}^{\mathbf{C}}_{i_r}\right)^H \mathbf{F}^{\mathbf{A}}_{i,j} \mathbf{V}^{\mathbf{B}}_{j_r} \mathbf{S}^{\mathbf{B}}_{j,k} \left(\mathbf{V}^{\mathbf{B}}_{k_c}\right)^T \left(\mathbf{V}^{\mathbf{C}}_{k_c}\right)^*, & \text{case-2} \\ \left(\mathbf{V}^{\mathbf{C}}_{i_r}\right)^H \mathbf{V}^{\mathbf{A}}_{i_r} \mathbf{S}^{\mathbf{A}}_{i,j} \left(\mathbf{V}^{\mathbf{A}}_{j_c}\right)^T \mathbf{F}^{\mathbf{B}}_{j,k} \left(\mathbf{V}^{\mathbf{C}}_{k_c}\right)^*, & \text{case-3} \\ \left(\mathbf{V}^{\mathbf{C}}_{i_r}\right)^H \mathbf{V}^{\mathbf{A}}_{i_r} \mathbf{S}^{\mathbf{A}}_{i,j} \mathbf{B}_j \mathbf{S}^{\mathbf{B}}_{j,k} \left(\mathbf{V}^{\mathbf{B}}_{k_c}\right)^T \left(\mathbf{V}^{\mathbf{C}}_{k_c}\right)^*, & \text{case-4.} \end{cases} \tag{17}$$

The resulting admissible blocks in $\mathbf{C}_{\mathcal{H}^2}$ are nothing but $\mathbf{R}^{\mathbf{C}}_{i,k} = \mathbf{V}^{\mathbf{C}}_{i_r} \times \mathbf{S}^{\mathbf{C}}_{i,k} \times (\mathbf{V}^{\mathbf{C}}_{k_c})^T$.

In (17), the $\mathbf{B}_j$ is the cluster bases product, which is as shown as follows:

$$\mathbf{B}_j = \left(\mathbf{V}^{\mathbf{A}}_{j_c}\right)^T \times \mathbf{V}^{\mathbf{B}}_{j_r}. \tag{18}$$

Since it is only related to the original cluster bases, it can be prepared in advance before the MMP computation. Using the nested property of the cluster bases, $\mathbf{B}_j$ can be computed in linear time for all clusters j, be j a leaf or a nonleaf cluster.

In (17), the $(\mathbf{V}^{\mathbf{C}}_{i_r})^H \mathbf{V}^{\mathbf{A}}_{i_r}$ is simply the projection of the original row cluster basis of $\mathbf{A}$ onto the new cluster basis of the product matrix $\mathbf{C}$. Similarly, $(\mathbf{V}^{\mathbf{B}}_{k_c})^T (\mathbf{V}^{\mathbf{C}}_{k_c})^*$ denotes the projection of the original column cluster basis of $\mathbf{B}$ onto the newly generated column cluster basis in $\mathbf{C}$. The two cluster basis projections can also be computed for every leaf cluster after the new cluster bases have been generated. Hence, we compute

$$\begin{aligned} \mathbf{P}^{\mathbf{A}}_i &= \left(\mathbf{V}^{\mathbf{C}}_{i_r}\right)^H \mathbf{V}^{\mathbf{A}}_{i_r} \\ \mathbf{P}^{\mathbf{B}}_k &= \left(\mathbf{V}^{\mathbf{B}}_{k_c}\right)^T \left(\mathbf{V}^{\mathbf{C}}_{k_c}\right)^* \end{aligned} \tag{19}$$

for each leaf row cluster i, and each column leaf cluster k. In this way, it can be reused without recomputing for each admissible block formed by i or k.

In (17), we can also see that the $\mathbf{F}$ block is front and back multiplied by cluster bases. It can be viewed as an $\mathbf{F}$ block collected based on the front (row) and back (column) cluster bases, which becomes a matrix of rank size. Specifically, in (17), there are three kinds of collected blocks

$$\begin{aligned} \left(\mathbf{F}^{\mathbf{A}}_{i,j} \mathbf{F}^{\mathbf{B}}_{j,k}\right)_{\text{coll.}} &= \left(\mathbf{V}^{\mathbf{C}}_{i_r}\right)^H \left(\mathbf{F}^{\mathbf{A}}_{i,j} \mathbf{F}^{\mathbf{B}}_{j,k}\right) \left(\mathbf{V}^{\mathbf{C}}_{k_c}\right)^* \\ \left(\mathbf{F}^{\mathbf{A}}_{i,j}\right)_{\text{coll.}} &= \left(\mathbf{V}^{\mathbf{C}}_{i_r}\right)^H \mathbf{F}^{\mathbf{A}}_{i,j} \mathbf{V}^{\mathbf{B}}_{j_r} \\ \left(\mathbf{F}^{\mathbf{B}}_{j,k}\right)_{\text{coll.}} &= \left(\mathbf{V}^{\mathbf{A}}_{j_c}\right)^T \mathbf{F}^{\mathbf{B}}_{j,k} \left(\mathbf{V}^{\mathbf{C}}_{k_c}\right)^* \end{aligned} \tag{20}$$

which is used in case-1–3 multiplication, respectively.

As can be seen from (17), the case-1 multiplication with an admissible block being the target can be performed by first computing the full-matrix product, and then collecting the product onto the new row and column cluster bases of the product matrix. This collect operation is accurate because the newly generated row and column cluster bases have taken such a case-1 multiplication into consideration when being generated. As for the case-2 multiplication, as can be seen from (17), we can use the $\mathbf{F}_{i,j}$ collected based on the new row cluster basis and the original column cluster basis, the size of which is rank, to multiply the coupling matrix of $\mathbf{S}_{j,k}$, and then multiply the column basis projection matrix since the column bases have been changed. Similarly, for case-3, we use the collected block $(\mathbf{F}^{\mathbf{B}}_{j,k})_{\text{coll.}}$, and front multiply it by the coupling matrix of $\mathbf{S}_{i,j}$, and then front multiply a row cluster basis transformation matrix. As for case-4, we multiply the coupling matrix of $\mathbf{A}$'s admissible block by the cluster basis product, and then by the coupling matrix of $\mathbf{B}$'s admissible block. Since the row and column cluster bases have been changed to account for the other cases of multiplications, at the end, we need to front and back multiply the cluster basis transformation matrices to complete the computation of case-4. Summarizing the aforementioned, the coupling matrix in (17) can be efficiently computed as

$$\mathbf{S}^{\mathbf{C}}_{i,k} = \begin{cases} \left(\mathbf{V}^{\mathbf{C}}_{i_r}\right)^H \mathbf{F}^{\mathbf{A}}_{i,j} \mathbf{F}^{\mathbf{B}}_{j,k} \left(\mathbf{V}^{\mathbf{C}}_{k_c}\right)^*, & \text{case-1} \\ \left(\mathbf{F}^{\mathbf{A}}_{i,j}\right)_{\text{coll.}} \mathbf{S}^{\mathbf{B}}_{j,k} \mathbf{P}^{\mathbf{B}}_k, & \text{case-2} \\ \mathbf{P}^{\mathbf{A}}_i \mathbf{S}^{\mathbf{A}}_{i,j} \left(\mathbf{F}^{\mathbf{B}}_{j,k}\right)_{\text{coll.}}, & \text{case-3} \\ \mathbf{P}^{\mathbf{A}}_i \mathbf{S}^{\mathbf{A}}_{i,j} \mathbf{B}_j \mathbf{S}^{\mathbf{B}}_{j,k} \mathbf{P}^{\mathbf{B}}_k, & \text{case-4.} \end{cases} \tag{21}$$

E. Summary of Overall Algorithm at Leaf Level

Here, we conclude all the operations related to leaf-level computation when the target is an admissible block.

1) Prepare cluster bases product $\mathbf{B}$.
2) Compute all the leaf-level row and column cluster bases of product matrix $\mathbf{C}_{\mathcal{H}^2}$.
3) Collect the $\mathbf{F}$ blocks in $\mathbf{A}_{\mathcal{H}^2}$ and $\mathbf{B}_{\mathcal{H}^2}$ based on the new row and/or column cluster bases, also prepare cluster bases transformation matrix $\mathbf{P}$.
4) Perform four cases of multiplications.

After leaf-level multiplications, we need to merge four coupling matrices at a nonleaf-level admissible block, as shown by the blue blocks in Fig. 3(c). These matrices correspond to the multiplication case of a nonleaf block $\mathbf{NL}$ multiplied by a nonleaf block $\mathbf{NL}$ generating an admissible block at next level. The merged block is the coupling matrix of this next-level admissible block. It will be used to update next-level transfer matrices. The details will be given in Section IV.

IV. Proposed $\mathcal{H}^2$ MMP Algorithm—Nonleaf Level

After finishing the leaf-level multiplication, we proceed to nonleaf-level multiplications. In Fig. 4, we use level $L-1$ as an example to illustrate $\mathbf{A}^{L-1}_{\mathcal{H}^2}$, $\mathbf{B}^{L-1}_{\mathcal{H}^2}$, and $\mathbf{C}^{L-1}_{\mathcal{H}^2}$.

At a nonleaf level l, there are also in total four matrix-matrix multiplication cases.

1) Case-1: $\mathbf{NL}^{\mathbf{A}} \times \mathbf{NL}^{\mathbf{B}}$.
2) Case-2: $\mathbf{NL}^{\mathbf{A}} \times \mathbf{R}^{\mathbf{B}}$.
3) Case-3: $\mathbf{R}^{\mathbf{A}} \times \mathbf{NL}^{\mathbf{B}}$.
4) Case-4: $\mathbf{R}^{\mathbf{A}} \times \mathbf{R}^{\mathbf{B}}$.

where $\mathbf{NL}$ denotes a nonleaf block. The resulting matrix block in $\mathbf{C}$ is also of two kinds: 1) nonleaf block $\mathbf{NL}$ at this level, marked in red in Fig. 4(c) and 2) admissible block $\mathbf{R}$, marked in green in Fig. 4(c). Next, we show how to perform each case of multiplications based on the two kinds of target blocks.

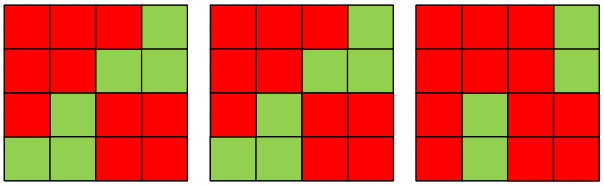


Fig. 4. $\mathcal{H}^2$-matrix block at nonleaf level $(L-1)$. (a) $\mathbf{A}_{\mathcal{H}^2}^{L-1}$. (b) $\mathbf{B}_{\mathcal{H}^2}^{L-1}$. (c) $\mathbf{C}_{\mathcal{H}^2}^{L-1}$.

A. Product is an NL Block in C

The **NL** target block would not exist for a case-1 multiplication, since if a case-1 multiplication results in an **NL** block, that computation should have been performed at previous level. As for the other three cases of multiplications, since at least one admissible block is present in the multipliers, the product must be an admissible block. Hence, we compute them as having an admissible block as the product, using the algorithm described in Section IV-B, and associate the resulting admissible block with the **NL** block. After the computation is done at all levels, we perform a backward split operation to split the admissible block associated with each **NL** block to each leaf block of **C** based on its structure.

B. Product is an Admissible Block in C

Similar to the leaf level, if the product is an admissible block in **C** whether at the same nonleaf level or at an upper level, case-4 can be performed as it is since the product matrix is obviously admissible, which also preserves the original row and column cluster bases. We can write

$$\begin{aligned}&\textbf{case-4:}\\&\mathbf{R}_{i,j}^{\mathbf{A}} \times \mathbf{R}_{j,k}^{\mathbf{B}}\\&= \mathbf{V}_{i_r^{\text{ch}}}^{\mathbf{A}} \mathbf{T}_{i_r}^{\mathbf{A}} \mathbf{S}_{i,j}^{\mathbf{A}} \left(\mathbf{T}_{j_c}^{\mathbf{A}}\right)^T \left(\mathbf{V}_{j_c^{\text{ch}}}^{\mathbf{A}}\right)^T \times \mathbf{V}_{j_r^{\text{ch}}}^{\mathbf{B}} \mathbf{T}_{j_r}^{\mathbf{B}} \mathbf{S}_{j,k}^{\mathbf{B}} \left(\mathbf{T}_{k_c}^{\mathbf{B}}\right)^T \left(\mathbf{V}_{k_c^{\text{ch}}}^{\mathbf{B}}\right)^T\end{aligned} \tag{22}$$

where **T** denotes a transfer matrix, and superscript *ch* denotes the two children clusters of the nonleaf cluster i. If the cluster bases at leaf level and the transfer matrices at nonleaf levels are kept the same as before, then the computation of (22) is to calculate the coupling matrix at level l, which is

$$\mathbf{S}_{i,k}^{\mathbf{C}} = \mathbf{S}_{i,j}^{\mathbf{A}}(\mathbf{B}_j)\mathbf{S}_{j,k}^{\mathbf{B}}. \tag{23}$$

It is a product of three small matrices whose size is the rank at this tree level. Rewriting (22) as

$$\begin{aligned}&\textbf{case-4:}\\&\mathbf{R}_{i,j}^{\mathbf{A}} \times \mathbf{R}_{j,k}^{\mathbf{B}} = \mathbf{V}_{i_r^{\text{ch}}}^{\mathbf{A}} \mathbf{T}_{i_r}^{\mathbf{A}} \mathbf{S}_{i,k}^{\mathbf{C}} \left(\mathbf{T}_{k_c}^{\mathbf{B}}\right)^T \left(\mathbf{V}_{k_c^{\text{ch}}}^{\mathbf{B}}\right)^T.\end{aligned} \tag{24}$$

If we exclude the children cluster bases in the front and at the back, we can see that **T** serves as the new cluster basis at this level. In other words, at a nonleaf level l, if we treat this level as the bottom level of the remaining tree, then the transfer matrix of the nonleaf cluster is nothing but the leaf cluster basis of the shortened tree.

Similar to the leaf-level computation, the other three cases of multiplications will result in a change of cluster basis in the matrix product. Specifically, case-1 results in a different row as well as column cluster bases in the product admissible block because

$$\textbf{case-1: } \mathbf{NL}_{i,j}^{\mathbf{A}} \times \mathbf{NL}_{j,k}^{\mathbf{B}} \tag{25}$$

case-2 yields a different row cluster basis since

$$\textbf{case-2: } \mathbf{NL}_{i,j}^{\mathbf{A}} \times \mathbf{R}_{j,k}^{\mathbf{B}} = \left(\mathbf{NL}_{i,j}^{\mathbf{A}} \mathbf{V}_{j_r}^{\mathbf{B}}\right) \times \mathbf{S}_{j,k}^{\mathbf{B}} \times \left(\mathbf{V}_{k_c}^{\mathbf{B}}\right)^T \tag{26}$$

whereas case-3 results in a different column cluster basis in the product admissible block, because

$$\textbf{case-3: } \mathbf{R}_{i,j}^{\mathbf{A}} \times \mathbf{NL}_{j,k}^{\mathbf{B}} = \mathbf{V}_{i_r}^{\mathbf{A}} \times \mathbf{S}_{i,j}^{\mathbf{A}} \times \left(\left(\mathbf{V}_{j_c}^{\mathbf{A}}\right)^T \mathbf{NL}_{j,k}^{\mathbf{B}}\right). \tag{27}$$

If we do not update the cluster bases in the product matrix, the accuracy of the multiplication is not controllable. However, if we update the cluster basis as they are, it is computationally very expensive since the matrix block size keeps increasing when we proceed from leaf level toward the root level. In addition to the cost of changing cluster bases, if we have to carry out the multiplications at each nonleaf level using the actual matrix block size, then the computation is also prohibitive. Therefore, the fast algorithm we develop here is to perform all computations using the rank size at each tree level, and meanwhile control the accuracy.

In the proposed algorithm, to account for the updates to the original matrix during the MMP procedure, the cluster bases of **C** are computed level by level, which are manifested by the changed leaf cluster bases and the transfer matrices at nonleaf levels. At a nonleaf level, its children-level cluster bases have already been computed, and they are different from the original ones in **A** and **B**. However, the new cluster bases have taken the upper-level multiplications into consideration. Hence, we can accurately represent the multiplication at the current nonleaf level using newly generated children cluster bases.

Take case-1 product as an example, where we perform $\mathbf{NL}_{i,j}^{\mathbf{A}} \times \mathbf{NL}_{j,k}^{\mathbf{B}}$ obtaining an admissible $\mathbf{R}_{i,k}^{\mathbf{C}}$. We can accurately represent this product using the children cluster bases of i and k as follows:

$$\begin{aligned}&\textbf{case-1:}\\&\mathbf{NL}_{i,j}^{\mathbf{A}} \times \mathbf{NL}_{j,k}^{\mathbf{B}}\\&= \begin{bmatrix} \mathbf{V}_{i1_r}^{\mathbf{C}} & \\ & \mathbf{V}_{i2_r}^{\mathbf{C}} \end{bmatrix} \left(\mathbf{NL}_{i,j}^{\mathbf{A}} \mathbf{NL}_{j,k}^{\mathbf{B}}\right)_{\text{coll.}} \begin{bmatrix} \left(\mathbf{V}_{k1_c}^{\mathbf{C}}\right)^T & \\ & \left(\mathbf{V}_{k2_c}^{\mathbf{C}}\right)^T \end{bmatrix}\end{aligned} \tag{28}$$

in which

$$\begin{aligned}&\left(\mathbf{NL}_{i,j}^{\mathbf{A}} \mathbf{NL}_{j,k}^{\mathbf{B}}\right)_{\text{coll.}}\\&= \begin{bmatrix} \left(\mathbf{V}_{i1_r}^{\mathbf{C}}\right)^H & \\ & \left(\mathbf{V}_{i2_r}^{\mathbf{C}}\right)^H \end{bmatrix} \left(\mathbf{NL}_{i,j}^{\mathbf{A}} \mathbf{NL}_{j,k}^{\mathbf{B}}\right) \begin{bmatrix} \left(\mathbf{V}_{k1_c}^{\mathbf{C}}\right)^* & \\ & \left(\mathbf{V}_{k2_c}^{\mathbf{C}}\right)^* \end{bmatrix}.\end{aligned} \tag{29}$$

This collected block, $(\mathbf{NL}_{i,j}^{\mathbf{A}} \mathbf{NL}_{j,k}^{\mathbf{B}})_{\text{coll.}}$, is actually the coupling matrix merged from the four small coupling matrices

computed at previous level, when dealing with the multiplication case of having a target block as a subblock in the upper-level admissible block. It can be written as

$$\left(\mathbf{NL}^{\mathbf{A}}_{i,j}\mathbf{NL}^{\mathbf{B}}_{j,k}\right)_{\text{coll.}} = \begin{bmatrix} \mathbf{S}^{\mathbf{C}}_{i_1,k_1} & \mathbf{S}^{\mathbf{C}}_{i_1,k_2} \\ \mathbf{S}^{\mathbf{C}}_{i_2,k_1} & \mathbf{S}^{\mathbf{C}}_{i_2,k_2} \end{bmatrix}. \tag{30}$$

Each of the four coupling matrices has been obtained at previous level. From (29), it is clear that using the nested property of the cluster bases, the collect operation does not need to start from leaf level, but using the four blocks obtained at previous one level.

For case-2 product, it can also be accurately expanded in the space of the children row cluster bases, and hence

$$\textbf{case-2: } \mathbf{NL}^{\mathbf{A}}_{i,j} \times \mathbf{R}^{\mathbf{B}}_{j,k} = \begin{bmatrix} \mathbf{V}^{\mathbf{C}}_{i1_r} & \\ & \mathbf{V}^{\mathbf{C}}_{i2_r} \end{bmatrix} \mathbf{NL}^{\mathbf{A}}_{i,j\,\text{coll.}} \mathbf{T}^{\mathbf{B}}_{j_r} \mathbf{S}^{\mathbf{B}}_{j,k} \left(\mathbf{V}^{\mathbf{B}}_{k_c}\right)^T \tag{31}$$

where

$$\mathbf{NL}^{\mathbf{A}}_{i,j\,\text{coll.}} = \begin{bmatrix} \left(\mathbf{V}^{\mathbf{C}}_{i1_r}\right)^H & \\ & \left(\mathbf{V}^{\mathbf{C}}_{i2_r}\right)^H \end{bmatrix} \mathbf{NL}^{\mathbf{A}}_{i,j} \begin{bmatrix} \mathbf{V}^{\mathbf{B}}_{j1_r} & \\ & \mathbf{V}^{\mathbf{B}}_{j2_r} \end{bmatrix} \tag{32}$$

which is $\mathbf{NL}^{\mathbf{A}}_{i,j}$ collected based on the children's new row cluster bases in $\mathbf{C}$ and the original column cluster bases in $\mathbf{B}$. From (31), it can be seen that if excluding the children cluster bases, then $\mathbf{NL}^{\mathbf{A}}_{i,j\,\text{coll.}}\mathbf{T}^{\mathbf{B}}_{j_r}$ resembles the $\mathbf{F}^{\mathbf{A}}_{i,j}\mathbf{V}^{\mathbf{B}}_{j_r}$ in the leaf level case-2 product. In other words, if we treat the current nonleaf level as the leaf level, then $\mathbf{NL}^{\mathbf{A}}_{i,j\,\text{coll.}}$ is equivalent to a full matrix block, whereas $\mathbf{T}$ is the leaf cluster basis. An example of $\mathbf{NL}^{\mathbf{A}}_{i,j\,\text{coll.}}$ block at level $(L-1)$ in $\mathbf{A}_{\mathcal{H}^2}$ can be seen as follows:

$$\left(\mathbf{NL}^{\mathbf{A}}_{i,j}\right)_{\text{coll.}} = \begin{bmatrix} \left(\mathbf{F}^{\mathbf{A}}_{i_1,j_1}\right)_{\text{coll.}} & \mathbf{P}^{\mathbf{A}}_{i_1}\mathbf{S}^{\mathbf{A}}_{i_1,j_2}\mathbf{B}_{j_2} \\ \mathbf{P}^{\mathbf{A}}_{i_2}\mathbf{S}^{\mathbf{A}}_{i_2,j_1}\mathbf{B}_{j_1} & \left(\mathbf{F}^{\mathbf{A}}_{i_2,j_2}\right)_{\text{coll.}} \end{bmatrix} \tag{33}$$

which consists of collected full matrices whose expressions are shown in (20), and projected coupling matrices of admissible blocks. Again, using the nested property of both new and original cluster bases, the collect operation does not need to start from leaf level, but using the four blocks obtained at previous one level. Each collect operation only costs $O(k_l)^3$, where k_l is the rank at level l.

Since the cluster bases at the previous level have been computed, for case-1 and case-2 products at a nonleaf level, we only need to compute the center block associated with the current nonleaf level, and this computation can be carried out in the same way as how we carry out leaf-level computation, if we treat the current nonleaf level as the leaf level of the remaining tree. The same is true to case-3 product, where we have

$$\textbf{case-3: } \mathbf{R}^{\mathbf{A}}_{i,j} \times \mathbf{NL}^{\mathbf{B}}_{j,k} = \mathbf{V}^{\mathbf{A}}_{i_r}\mathbf{S}^{\mathbf{A}}_{i,j}\left(\mathbf{T}^{\mathbf{A}}_{j_c}\right)^T \mathbf{NL}^{\mathbf{B}}_{j,k\,\text{coll.}} \begin{bmatrix} \left(\mathbf{V}^{\mathbf{C}}_{k1_c}\right)^T & \\ & \left(\mathbf{V}^{\mathbf{C}}_{k2_c}\right)^T \end{bmatrix} \tag{34}$$

in which

$$\mathbf{NL}^{\mathbf{B}}_{j,k\,\text{coll.}} = \begin{bmatrix} \left(\mathbf{V}^{\mathbf{A}}_{j1_r}\right)^T & \\ & \left(\mathbf{V}^{\mathbf{A}}_{j2_r}\right)^T \end{bmatrix} \mathbf{NL}^{\mathbf{B}}_{j,k} \begin{bmatrix} \left(\mathbf{V}^{\mathbf{C}}_{k1_c}\right)^* & \\ & \left(\mathbf{V}^{\mathbf{C}}_{k2_c}\right)^* \end{bmatrix}. \tag{35}$$

We can see that $(\mathbf{T}^{\mathbf{A}}_{j_c})^T\mathbf{NL}^{\mathbf{B}}_{j,k\,\text{coll.}}$ resembles the $(\mathbf{V}^{\mathbf{A}}_{j_c})^T\mathbf{F}^{\mathbf{B}}_{j,k}$ in the leaf-level case-3 product. An example of collected $\mathbf{NL}$ block in $\mathbf{B}_{\mathcal{H}^2}$ is given as follows:

$$\left(\mathbf{NL}^{\mathbf{B}}_{i,j}\right)_{\text{coll.}} = \begin{bmatrix} \left(\mathbf{F}^{\mathbf{B}}_{i_1,j_1}\right)_{\text{coll.}} & \mathbf{B}_{i_1}\mathbf{S}^{\mathbf{B}}_{i_1,j_2}\mathbf{P}^{\mathbf{B}}_{j_2} \\ \mathbf{B}_{i_2}\mathbf{S}^{\mathbf{B}}_{i_2,j_1}\mathbf{P}^{\mathbf{B}}_{j_1} & \left(\mathbf{F}^{\mathbf{B}}_{i_2,j_2}\right)_{\text{coll.}} \end{bmatrix} \tag{36}$$

which consists of collected full matrices whose expressions are shown in (20), and projected coupling matrices of admissible blocks.

Since the cluster bases have been changed at previous level, we also represent the case-4 product using the new children cluster bases of i and k, thus

$$\textbf{case-4: } \mathbf{R}^{\mathbf{A}}_{i,j} \times \mathbf{R}^{\mathbf{B}}_{j,k} = \begin{bmatrix} \left(\mathbf{V}^{\mathbf{C}}_{i1_r}\right) & \\ & \left(\mathbf{V}^{\mathbf{C}}_{i2_r}\right) \end{bmatrix} \mathbf{R}^{\mathbf{C}}_{i,k,\text{proj}} \begin{bmatrix} \left(\mathbf{V}^{\mathbf{C}}_{k1_c}\right)^T & \\ & \left(\mathbf{V}^{\mathbf{C}}_{k2_c}\right)^T \end{bmatrix} \tag{37}$$

and

$$\mathbf{R}^{\mathbf{C}}_{i,k,\text{proj}} = \begin{bmatrix} \left(\mathbf{P}^{\mathbf{A}}_{i1}\right) & \\ & \left(\mathbf{P}^{\mathbf{A}}_{i2}\right) \end{bmatrix} \left(\mathbf{T}^{\mathbf{A}}_{i_r}\mathbf{S}^{\mathbf{C}}_{i,k}\left(\mathbf{T}^{\mathbf{B}}_{k_c}\right)^T\right) \begin{bmatrix} \left(\mathbf{P}^{\mathbf{B}}_{k1}\right) & \\ & \left(\mathbf{P}^{\mathbf{B}}_{k2}\right) \end{bmatrix} \tag{38}$$

which can be written in short as

$$\mathbf{R}^{\mathbf{C}}_{i,k,\text{proj}} = \mathbf{P}^{\mathbf{A}}_{i^{\text{ch}}}\left(\mathbf{T}^{\mathbf{A}}_{i_r}\mathbf{S}^{\mathbf{C}}_{i,k}\left(\mathbf{T}^{\mathbf{B}}_{k_c}\right)^T\right)\mathbf{P}^{\mathbf{B}}_{k^{\text{ch}}} \tag{39}$$

where ch denotes children. Here, there is a cluster basis transformation matrix in the front and at the back.

C. Computation of the New Nonleaf-Level Transfer Matrices in C

If the target block is an admissible block at a nonleaf level, we need to represent it as $\mathbf{R}_{t,s} = \mathbf{T}_t\mathbf{S}_{t,s}(\mathbf{T}_s)^T$ in controlled accuracy. Hence, we need to calculate new row and column transfer matrices $\mathbf{T}$ of product matrix $\mathbf{C}_{\mathcal{H}^2}$. First, we introduce how to calculate the row transfer matrices. Similar to leaf level, case-1 and 2 products result in a change in the row cluster basis and hence row transfer matrix. Case-3 and 4 products do not require a change of transfer matrix if the cluster bases have not been changed at previous level. However, since the cluster bases have been changed at previous level, the transfer matrix requires an update as well.

For an arbitrary nonleaf cluster i, we first find all of the case-1 products associated with i. Each of such a product leads to a coupling matrix merged from the four coupling matrices obtained at previous level computation, denoted by $(\mathbf{NL}^{\mathbf{A}}_{i,j}\mathbf{NL}^{\mathbf{B}}_{j,k})_{\text{coll.}}$. Using them, we calculate the Gram matrix sum as

$$\mathbf{G}^{\mathbf{C},l}_{i_{r1}} = \sum_{\#(i,k)=1}^{O\left(C^2_{\text{sp}}\right)} \left(\mathbf{NL}^{\mathbf{A}}_{i,j}\mathbf{NL}^{\mathbf{B}}_{j,k}\right)_{\text{coll.}} \left(\left(\mathbf{NL}^{\mathbf{A}}_{i,j}\mathbf{NL}^{\mathbf{B}}_{j,k}\right)_{\text{coll.}}\right)^H. \tag{40}$$

The second step is to take case-2 multiplications at a nonleaf level into consideration for row transfer matrix calculation of product matrix $\mathbf{C}_{\mathcal{H}^2}$. We find all the collected nonleaf blocks $\mathbf{NL}^{\mathbf{A}}_{i,j\,\text{coll.}}$ of cluster i at level l in $\mathbf{A}_{\mathcal{H}^2}$ matrix and multiply them with corresponding transfer matrices $\mathbf{T}^{\mathbf{B}}_{j_r}$ from $\mathbf{B}_{\mathcal{H}^2}$ matrix. And we calculate the Gram matrix sum as

$$\mathbf{G}^{\mathbf{C},l}_{i_{r2}} = \sum_{j=1}^{O(C_{\text{sp}})} \left(\left(\mathbf{NL}^{\mathbf{A}}_{i,j}\right)_{\text{coll.}}\mathbf{T}^{\mathbf{B}}_{j_r}\right)\left(\left(\mathbf{NL}^{\mathbf{A}}_{i,j}\right)_{\text{coll.}}\mathbf{T}^{\mathbf{B}}_{j_r}\right)^H. \quad (41)$$

Finally, we count the contributions from case-3 and case-4 products by computing

$$\mathbf{G}^{\mathbf{C},l}_{i_{r3}} = \mathbf{P}^{\mathbf{A}}_{i^{\text{ch}}}\mathbf{T}^{\mathbf{A}}_{i_r}\left(\mathbf{T}^{\mathbf{A}}_{i_r}\right)^H\left(\mathbf{P}^{\mathbf{A}}_{i^{\text{ch}}}\right)^H. \quad (42)$$

Again, we normalize these three Gram matrices and obtain

$$\mathbf{G}^{\mathbf{C},l}_{i_r} = \widehat{\mathbf{G}^{\mathbf{C},l}_{i_{r1}}} + \widehat{\mathbf{G}^{\mathbf{C},l}_{i_{r2}}} + \widehat{\mathbf{G}^{\mathbf{C},l}_{i_{r3}}}. \quad (43)$$

We then calculate an SVD of this $\mathbf{G}^{\mathbf{C},l}_{i_r}$ and truncate the singular values based on prescribed accuracy ϵ_{trunc} to obtain row transfer matrix $\mathbf{T}^{\mathbf{C}}_{i_r}$ for cluster i at nonleaf level. Notice that the size of $\mathbf{G}^{\mathbf{C},l}_{i_r}$ is rank k_l, and hence the SVD's cost is only $O(k_l^3)$.

Similarly, we can compute the new column transfer matrices for nonleaf cluster k, which is $\mathbf{T}^{\mathbf{C}}_{k_c}$. The first part is

$$\mathbf{G}^{\mathbf{C},l}_{k_{c1}} = \sum_{i=1}^{O(C_{\text{sp}})} \left(\left(\mathbf{NL}^{\mathbf{A}}_{i,j}\mathbf{NL}^{\mathbf{B}}_{j,k}\right)_{\text{coll.}}\right)^T\left(\left(\mathbf{NL}^{\mathbf{A}}_{i,j}\mathbf{NL}^{\mathbf{B}}_{j,k}\right)_{\text{coll.}}\right)^*. \quad (44)$$

The second part is

$$\mathbf{G}^{\mathbf{C},l}_{k_{c2}} = \sum_{j=1}^{O(C_{\text{sp}})}\left(\left(\mathbf{T}^{\mathbf{A}}_{j_c}\right)^T\left(\mathbf{NL}^{\mathbf{B}}_{j,k}\right)_{\text{coll.}}\right)^T\left(\left(\mathbf{T}^{\mathbf{A}}_{j_c}\right)^T\left(\mathbf{NL}^{\mathbf{B}}_{j,k}\right)_{\text{coll.}}\right)^*. \quad (45)$$

The third part is

$$\mathbf{G}^{\mathbf{C},l}_{k_{c3}} = \left(\mathbf{P}^{\mathbf{B}}_{k^{\text{ch}}}\right)^T\left(\mathbf{T}^{\mathbf{B}}_{k_c}\right)^*\left(\mathbf{T}^{\mathbf{B}}_{k_c}\right)^T\left(\mathbf{P}^{\mathbf{B}}_{k^{\text{ch}}}\right)^*. \quad (46)$$

Then we normalize the three Gram matrices and sum them up as

$$\mathbf{G}^{\mathbf{C},l}_{k_c} = \widehat{\mathbf{G}^{\mathbf{C},l}_{k_{c1}}} + \widehat{\mathbf{G}^{\mathbf{C},l}_{k_{c2}}} + \widehat{\mathbf{G}^{\mathbf{C},l}_{k_{c3}}}. \quad (47)$$

After we perform an SVD on $\mathbf{G}^{\mathbf{C},l}_{k_c}$ matrix and truncate the singular values based on prescribed accuracy ϵ_{trunc}, we get new column transfer matrix $\mathbf{T}^{\mathbf{C}}_{k_c}$. Again, notice that the size of $\mathbf{G}^{\mathbf{C},l}_{k_c}$ is rank k_l, and hence the SVD's cost is only $O(k_l^3)$.

D. Computation of the Four Cases of Multiplications With the Product Block Being Admissible

Now, we obtain both row and column transfer matrices for product matrix $\mathbf{C}_{\mathcal{H}^2}$; hence, the four multiplications become the computation of the coupling matrices, so that the admissible block at the current level has a form of $\mathbf{R}_{t,s} = \mathbf{T}_t\mathbf{S}_{t,s}(\mathbf{T}_s)^T$. The coupling matrix $\mathbf{S}$'s calculation is similar to that of leaf level in (17), which has the following expressions:

$$\mathbf{S}^{\mathbf{C}}_{i,k} = \begin{cases} \left(\mathbf{T}^{\mathbf{C}}_{i_r}\right)^H\left(\mathbf{NL}^{\mathbf{A}}_{i,j}\mathbf{NL}^{\mathbf{B}}_{j,k}\right)_{\text{coll.}}\left(\mathbf{T}^{\mathbf{C}}_{k_c}\right)^*, & \text{case-1} \\ \left(\mathbf{T}^{\mathbf{C}}_{i_r}\right)^H\left(\mathbf{NL}^{\mathbf{A}}_{i,j}\right)_{\text{coll.}}\mathbf{T}^{\mathbf{B}}_{j_r}\mathbf{S}^{\mathbf{B}}_{j,k}\left(\mathbf{V}^{\mathbf{B}}_{k_c}\right)^T\left(\mathbf{V}^{\mathbf{C}}_{k_c}\right)^*, & \text{case-2} \\ \left(\mathbf{V}^{\mathbf{C}}_{i_r}\right)^H\mathbf{V}^{\mathbf{A}}_{i_r}\mathbf{S}^{\mathbf{A}}_{i,j}\left(\mathbf{T}^{\mathbf{A}}_{j_c}\right)^T\left(\mathbf{NL}^{\mathbf{B}}_{j,k}\right)_{\text{coll.}}\left(\mathbf{T}^{\mathbf{C}}_{k_c}\right)^*, & \text{case-3} \\ \left(\mathbf{V}^{\mathbf{C}}_{i_r}\right)^H\mathbf{V}^{\mathbf{A}}_{i_r}\mathbf{S}^{\mathbf{A}}_{i,j}\mathbf{B}_j\mathbf{S}^{\mathbf{B}}_{j,k}\left(\mathbf{V}^{\mathbf{B}}_{k_c}\right)^T\left(\mathbf{V}^{\mathbf{C}}_{k_c}\right)^*, & \text{case-4.} \end{cases} \quad (48)$$

Again, we should prepare some matrix products in advance so that we can achieve linear complexity MMP for constant rank $\mathcal{H}^2$-matrix. For nonleaf levels, the cluster bases product $\mathbf{B}_j$ can be readily calculated using children's cluster bases based on the nested property. For example, given a nonleaf cluster j, we can generate $\mathbf{B}_j$ by using the cluster bases product of its children clusters j_1 and j_2, which is shown as

$$\mathbf{B}_j = \left(\mathbf{T}^{\mathbf{A}}_{j_{1c}}\right)^T\mathbf{B}_{j_1}\mathbf{T}^{\mathbf{B}}_{j_{1r}} + \left(\mathbf{T}^{\mathbf{A}}_{j_{2c}}\right)^T\mathbf{B}_{j_2}\mathbf{T}^{\mathbf{B}}_{j_{2r}}. \quad (49)$$

In addition, since the cluster bases product $\mathbf{B}_j$ only involve original cluster bases in $\mathbf{A}_{\mathcal{H}^2}$ and $\mathbf{B}_{\mathcal{H}^2}$ matrices, we can prepare the abovementioned $\mathbf{B}_j$ for all leaf and nonleaf clusters before MMP algorithm. In addition, the nonleaf-level cluster bases projection (transformation) can also be calculated using children's ones as shown in (19). The formulas are given as follows:

$$\begin{aligned} \mathbf{P}^{\mathbf{A}}_i &= \left(\mathbf{V}^{\mathbf{C}}_{i_r}\right)^H\mathbf{V}^{\mathbf{A}}_{i_r} \\ &= \left(\mathbf{T}^{\mathbf{C}}_{i_{1r}}\right)^H\mathbf{P}^{\mathbf{A}}_{i_1}\mathbf{T}^{\mathbf{A}}_{i_{1r}} + \left(\mathbf{T}^{\mathbf{C}}_{i_{2r}}\right)^H\mathbf{P}^{\mathbf{A}}_{i_2}\mathbf{T}^{\mathbf{A}}_{i_{2r}} \\ \mathbf{P}^{\mathbf{B}}_k &= \left(\mathbf{V}^{\mathbf{B}}_{k_c}\right)^T\left(\mathbf{V}^{\mathbf{C}}_{k_c}\right)^* \\ &= \left(\mathbf{T}^{\mathbf{B}}_{k_{1c}}\right)^T\mathbf{P}^{\mathbf{B}}_{k_1}\left(\mathbf{T}^{\mathbf{C}}_{k_{1c}}\right)^* + \left(\mathbf{T}^{\mathbf{B}}_{k_{2c}}\right)^T\mathbf{P}^{\mathbf{B}}_{k_2}\left(\mathbf{T}^{\mathbf{C}}_{k_{2c}}\right)^*. \end{aligned} \quad (50)$$

We also compute the collected $\mathbf{NL}$ matrix block in $\mathbf{A}_{\mathcal{H}^2}$ and $\mathbf{B}_{\mathcal{H}^2}$ at current level l by the following equation:

$$\begin{aligned} \left(\mathbf{NL}^{\mathbf{A}}_{i,j}\right)^{(l)}_{\text{coll.}} &= \left(\mathbf{T}^{\mathbf{C}}_{i_r}\right)^H\left(\mathbf{NL}^{\mathbf{A}}_{i,j}\right)^{(l+1)}_{\text{coll.}}\mathbf{T}^{\mathbf{B}}_{j_r} \\ \left(\mathbf{NL}^{\mathbf{B}}_{j,k}\right)^{(l)}_{\text{coll.}} &= \left(\mathbf{T}^{\mathbf{A}}_{j_c}\right)^T\left(\mathbf{NL}^{\mathbf{B}}_{j,k}\right)^{(l+1)}_{\text{coll.}}\left(\mathbf{T}^{\mathbf{C}}_{k_c}\right)^* \end{aligned} \quad (51)$$

where superscript l denotes tree level. After we prepare the matrix products in (49)–(51), we can proceed to calculate the coupling matrices in (48) efficiently as

$$\mathbf{S}^{\mathbf{C}}_{i,k} = \begin{cases} \left(\mathbf{T}^{\mathbf{C}}_{i_r}\right)^H\left(\mathbf{NL}^{\mathbf{A}}_{i,j}\mathbf{NL}^{\mathbf{B}}_{j,k}\right)^{(l)}_{\text{coll.}}\left(\mathbf{T}^{\mathbf{C}}_{k_c}\right)^*, & \text{case-1} \\ \left(\mathbf{NL}^{\mathbf{A}}_{i,j}\right)^{(l)}_{\text{coll.}}\mathbf{S}^{\mathbf{B}}_{j,k}\mathbf{P}^{\mathbf{B}}_k, & \text{case-2} \\ \mathbf{P}^{\mathbf{A}}_i\mathbf{S}^{\mathbf{A}}_{i,j}\left(\mathbf{NL}^{\mathbf{B}}_{j,k}\right)^{(l)}_{\text{coll.}}, & \text{case-3} \\ \mathbf{P}^{\mathbf{A}}_i\mathbf{S}^{\mathbf{A}}_{i,j}\mathbf{B}_j\mathbf{S}^{\mathbf{B}}_{j,k}\mathbf{P}^{\mathbf{B}}_k, & \text{case-4.} \end{cases} \quad (52)$$

All the coupling matrices calculation are performed in rank size k_l. So the computational cost is $O(k_l^3)$. After coupling matrices calculation in (52), all the admissible products at this nonleaf-level multiplication can be represented as $\mathbf{R}^{\mathbf{C}}_{i,j} = \mathbf{T}^{\mathbf{C}}_{i_r}\mathbf{S}^{\mathbf{C}}_{i,j}\mathbf{T}^{\mathbf{C}}_{j_c}$.

E. Summary of Overall Algorithm at Each Nonleaf Level

The cluster bases products $\mathbf{B}_j$ have been computed for all clusters j before the MMP starts, since they are only related to the original cluster bases.

At each nonleaf level, we do the following.

1) Collect four blocks in an **NL** block in $\mathbf{A}_{\mathcal{H}^2}$ to a block of $O(k_{l+1})$ size, using the newly generated children row cluster bases of **C** (transfer matrices if children are not at the leaf level) and the original column cluster bases of **B** (or transfer matrices). This is to generate the $(\mathbf{NL}^{\mathbf{A}}_{i,j})_{\text{coll.}}$, shown in (32).
2) Collect four blocks in an **NL** block in $\mathbf{B}_{\mathcal{H}^2}$ to a block of $O(k_{l+1})$ size, using the original row cluster bases of **A** (or transfer matrices) and the new children column cluster bases of **C** (transfer matrices if children are not at the leaf level). This is to generate $(\mathbf{NL}^{\mathbf{B}}_{j,k})_{\text{coll.}}$, shown in (35).
3) Merge four blocks in an **R** block in $\mathbf{C}_{\mathcal{H}^2}$. This corresponds to the $(\mathbf{NL}^{\mathbf{A}}_{i,j}\mathbf{NL}^{\mathbf{B}}_{j,k})^{(l)}_{\text{coll.}}$ in (52).
4) Calculate new row and column transfer matrices of product matrix $\mathbf{C}_{\mathcal{H}^2}$ at this level.
5) Prepare cluster bases projections $\mathbf{P}^{\mathbf{A}}_i$, $\mathbf{P}^{\mathbf{B}}_k$, and perform an **NL** block collect shown in (51).
6) Perform four cases of multiplications shown in (52).

After we finish one-way bottom-up tree traversal to calculate block matrix products at all the levels, i.e., from leaf level all the way up to minimal admissible level, we need to perform a postprocessing for the coupling matrices associated with the **NL** blocks in $\mathbf{C}_{\mathcal{H}^2}$. They exist because of the multiplications cases described in Section IV-A. This could be efficiently done by performing one-way top-down split process, the same as the matrix backward transformation shown in [1]. This postprocessing stage is to split the coupling matrices in **NL** to lower level admissible or inadmissible blocks.

V. Eight by Eight MMP Example

To help understand the overall algorithm, here, we use an eight by eight MMP as an example to explain the computation step by step. The matrices **A** and **B** shown in Fig. 2 can each be viewed as a matrix of size eight. Now, we show how these two matrices are multiplied to obtain **C**, as shown in Fig. 2. Each matrix is partitioned into green (admissible) blocks, and red (inadmissible) ones. The partition is performed using the admissibility condition given in (1). For this example, we can see that there are eight leaf clusters, and the minimal level that has admissible blocks is $L-1$, i.e., one level above the leaf level L. This is because the largest admissible block size is two. There are no green blocks whose size is four or larger. Therefore, the computation only needs to be performed from leaf level to $L-1$, the minimal level that has admissible blocks. Hence, in this simple example, there are only two levels of computation.

We start from the leaf level, and compute all the matrix multiplications whose matrix size is the $leafsize$, which is one in this example. In other words, we find all the leafblocks in **A**, and multiply them by the corresponding leafblocks in **B**. These blocks are shown in Fig. 3(a) and (b). For example, $\mathbf{A}_{1,j}$ $(j = 1, 2, \ldots, 6)$ are leaf blocks in **A** formed by leaf cluster $i = 1$ with other clusters, out of which $\mathbf{A}_{1,1}$, $\mathbf{A}_{1,2}$, $\mathbf{A}_{1,4}$, $\mathbf{A}_{1,6}$ are inadmissible; and the rest are admissible. Notice that the number of both kinds of blocks formed by a single cluster is bounded by constant C_{sp}. Similarly, in **B**, $\mathbf{B}_{1,j}$ $(j = 1, 2, \ldots, 6)$ are all the leaf blocks formed by leaf cluster $i = 1$; $\mathbf{B}_{3,j}$ $(j = 1, 2, \ldots, 4)$ are leaf blocks formed by leaf cluster $i = 3$, and so on. Notice that in the two figures, the white blocks are not zero, but admissible blocks at level $L-1$, and hence they are not involved in the leaf-level multiplication.

Next, we proceed to perform $\mathbf{A} \times \mathbf{B}$ at the leaf level. This is to compute all $\mathbf{A}_{i,j} \times \mathbf{B}_{j,k}$ that exists at this level, and store the result in $\mathbf{C}_{i,k}$. Therefore, we go through every leaf cluster i, and finish up the computation of all $\mathbf{A}_{i,j} \times \mathbf{B}_{j,k}$, and then the multiplication is done at the leaf level. Since for each cluster i, it can only form C_{sp} blocks with other clusters; there are at most C_{sp} $\mathbf{A}_{i,j}$ blocks; similarly, for each cluster j, it can only form C_{sp} blocks with other clusters; there are at most C_{sp} $\mathbf{B}_{j,k}$. Hence, for each cluster, the number of multiplications to do is bounded by C^2_{sp}, a constant.

$\mathbf{A}_{i,j}$ can only fall into two kinds, one is a full matrix $\mathbf{F}^{\mathbf{A}}_{i,j}$ (red block), and the other is admissible $\mathbf{R}^{\mathbf{A}}_{i,j}$ (green block). For any $\mathbf{F}^{\mathbf{A}}_{i,j}$, it gets multiplied by a leaf block $\mathbf{B}_{j,k}$, which is also either full $\mathbf{F}^{\mathbf{B}}_{j,k}$ or admissible $\mathbf{R}^{\mathbf{B}}_{j,k}$. This corresponds to the case-1 and case-2 products shown in Section III. Similarly, for every $\mathbf{R}^{\mathbf{A}}_{i,j}$, it gets multiplied by either $\mathbf{F}^{\mathbf{B}}_{j,k}$ or admissible $\mathbf{R}^{\mathbf{B}}_{j,k}$. This corresponds to the case-3 and case-4 products shown in Section III.

If the target block, $\mathbf{C}_{i,k}$, is inadmissible, we simply multiply the leaf blocks from **A** and **B** as they are. However, if the $\mathbf{C}_{i,k}$ is admissible, then we need to update the row and column cluster bases of $\mathbf{C}_{i,k}$ to ensure the accuracy of the multiplication. This is because case-1 and case-2 products will change the row cluster bases of admissible $\mathbf{C}_{i,k}$ block, i.e., we cannot use the original cluster bases of i any more to represent $\mathbf{C}_{i,k}$; and similarly, case-1 and case-3 products will change the column cluster bases of admissible $\mathbf{C}_{i,k}$. The change of cluster bases is done in the proposed algorithm in a systematical way. Basically, for each cluster i, we consider the contributions from all the case-1 multiplications of $\mathbf{F}^{\mathbf{A}}_{i,j}\mathbf{F}^{\mathbf{B}}_{j,k}$, and those from case-2 multiplications using $\mathbf{F}^{\mathbf{A}}_{i,j}\mathbf{V}^{\mathbf{B}}_{j}$, and use them to update the row cluster basis of cluster i in the product matrix, as shown in (12). Since the number of these multiplications is bounded by $O(C^2_{\text{sp}})$, a constant, the cost is small for updating the row cluster bases. We update the column cluster bases in a similar way. After updating the cluster bases at the leaf level, the admissible $\mathbf{C}_{i,k}$ can be accurately represented by the new leaf-level cluster bases. The remaining computation is simply to compute the four cases of multiplications based on the new bases. This is to compute the new coupling matrix of each admissible target, as shown in (17).

Then we move to the level $L-1$. The blocks at this level are shown in Fig. 4, where a red block denotes a nonleaf block (block composed of both inadmissible and admissible subblocks), and a green one is admissible. Each block is of size two now, i.e., two times of the leaf block size. As can be seen, there are now four clusters at this level, each of which is a nonleaf cluster of size two. The multiplications at this level are again to compute all $\mathbf{A}_{i,j} \times \mathbf{B}_{j,k}$ that exists at this level, and store the result in $\mathbf{C}_{i,k}$ at the same level. Out of the three blocks, one of them must be admissible for the

multiplication to be performed; otherwise, i.e., if all of them are nonleaf, then the multiplication has already been done at the children level of this level. So for every cluster i at level $L-1$, we use $\mathbf{A}_{i,j}$ that exists at this level, which can be either an $\mathbf{NL}^{\mathbf{A}}_{i,j}$ or an $\mathbf{R}^{\mathbf{A}}_{i,j}$, to multiply $\mathbf{B}_{j,k}$ at the same level, which also can be either an $\mathbf{NL}^{\mathbf{B}}_{j,k}$ or a $\mathbf{R}^{\mathbf{B}}_{j,k}$. This results in the four cases of multiplications discussed in Section IV. Apparently the matrix size is doubled, the computation must be performed on a double-sized matrix. In fact, using the proposed algorithm, the multiplication is performed on a rank size for each level. Take $\mathbf{NL}^{\mathbf{A}}_{i,j} \times \mathbf{R}^{\mathbf{B}}_{j,k}$ as an example, this product can be accurately represented using the row cluster bases made of the two children's cluster bases of i. This is because the two children cluster bases of i have already been updated to account for such a multiplication. To see this point clearly, one should realize that the $\mathbf{NL}^{\mathbf{A}}_{i,j}$ is composed of four subblocks formed between i's two children clusters, and j's two children clusters. Similarly, $\mathbf{R}^{\mathbf{B}}_{j,k}$ is also made of four subblocks. The multiplications between $\mathbf{NL}^{\mathbf{A}}_{i,j}$'s four subblocks and $\mathbf{R}^{\mathbf{B}}_{j,k}$'s four subblocks have been used to change the row cluster bases of $i1$ and $i2$, the i's two children cluster. Therefore, the multiplication of the $\mathbf{NL}^{\mathbf{A}}_{i,j}\mathbf{R}^{\mathbf{B}}_{j,k}$ becomes using the $\mathbf{NL}^{\mathbf{A}}_{i,j\,\text{coll.}}$ which is of rank size, obtained from collecting $\mathbf{NL}^{\mathbf{A}}_{i,j}$ block using the two children's new row cluster bases of cluster i and the original column cluster bases of j as shown in (32), to multiply the coupling matrix of $\mathbf{R}_{j,k}$, which is also of rank size, to obtain the coupling matrix of the target admissible block.

For the example shown in Fig. 4, take nonleaf cluster $i = 2$ as an example, it forms four blocks: $\mathbf{A}_{2,1}$ is an **NL** block; $\mathbf{A}_{2,2}$ is an **NL** block; and $\mathbf{A}_{2,3}$ and $\mathbf{A}_{2,4}$ are **R** blocks. The $\mathbf{A}_{2,1}$ will be multipled by $\mathbf{B}_{1,1}$ (**NL** block), $\mathbf{B}_{1,2}$ (**NL** block), $\mathbf{B}_{1,3}$ (**NL** block), and $\mathbf{B}_{1,4}$ (**R** block), respectively. However, only $\mathbf{A}_{2,1} \times \mathbf{B}_{1,4}$ need to be computed since its target block $\mathbf{A}_{2,4}$ is green, whereas other three multiplications are all an **NL** multiplied by an **NL** generating an **NL**, which has been computed using the children-level blocks already. We finish all the block multiplication at this level, each of which is one of the four cases shown in Section IV, then the computation is done at this level.

The end result of the MMP is **C**, which is stored as an $\mathcal{H}^2$-matrix, but its cluster bases have been changed based on accuracy, and the coupling matrix of each admissible block has been obtained. The inadmissible blocks are also computed.

VI. Accuracy and Complexity Analysis

In this section, we analyze the accuracy and computational complexity of the proposed algorithm to compute $\mathcal{H}^2$-MMPs.

A. Accuracy

Different from existing formatted $\mathcal{H}^2$-MMPs [1], in the proposed new algorithm, the accuracy of the product is directly controlled by ϵ_{trunc}. No formatted multiplications are performed, and the cluster bases are changed to represent the updates to the original matrix accurately. This makes each operation performed in the proposed MMP controlled by accuracy or exact. When generating an $\mathcal{H}^2$-matrix to represent the original dense matrix, the accuracy is controlled by $\epsilon_{\mathcal{H}^2}$, which is the same as in [15] and [16].

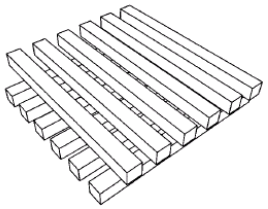


Fig. 5. Illustration of the 2-layer bus structure [7], where the number of wires in each layer, m, ranges from 8 to 128.

B. Time and Memory Complexity

The proposed MMP involves $O(L)$ levels of computation. At each level, there are 2^l clusters. For each cluster, the cost of changing the cluster bases at the leaf level due to four cases of multiplications is to perform $O(C_{\text{sp}})^2$ multiplications followed by an SVD, and each of which has a constant cost, as can be seen from (13), (14), and (15). The cost of changing the cluster bases at the nonleaf level due to the four cases of multiplications is also to perform $O(C_{\text{sp}})^2$ multiplications for each cluster followed by an SVD, and each of which has a cost of $O(k_l)^3$, as can be seen from (40)–(42). Notice that the **NL** blocks in **A** and **B** are collected level by level, at each level, there are $2^l O(C_{\text{sp}})$ **NL** block, and each collect operation also costs $O(k_l)^3$ only. Other auxiliary matrices are generated using a similar computational cost. The SVD is performed on a matrix of $leafsize$ for each cluster at the leaf level, and hence the cost is a constant. At a nonleaf level l, the SVD is performed on a matrix of rank size k_l for each cluster, and hence the cost is $O(k_l)^3$.

As for the computation of the four cases of multiplications at each level, each case involves $O(C_{\text{sp}})^2$ multiplications for each cluster, and each of which costs $O(k_l)^3$ at the nonleaf level and $O(leafsize)^3$ at the leaf level as can be seen from (21), and (52).

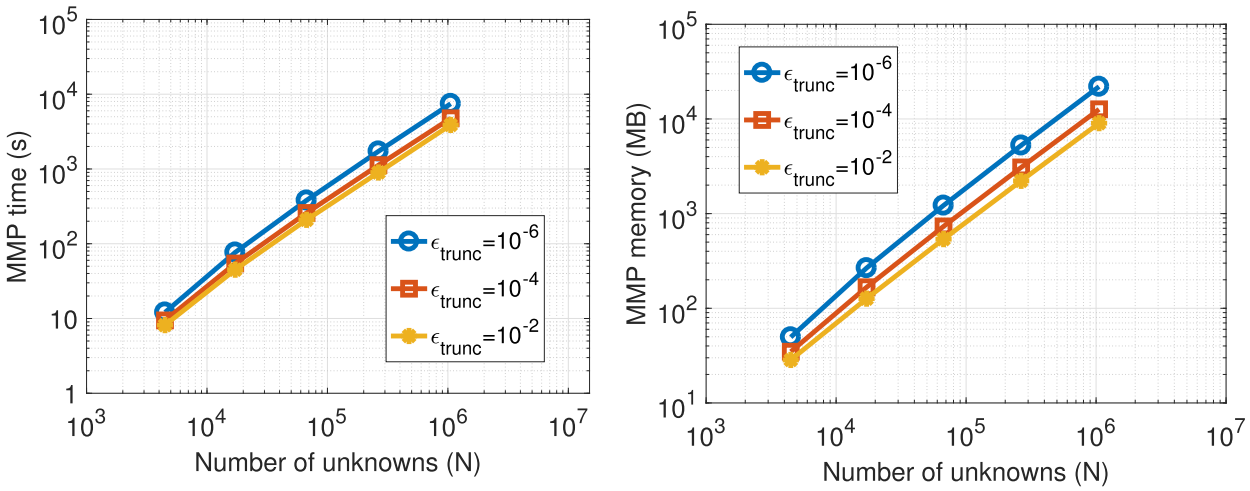
Hence, the time complexity of the proposed MMP can be found as

$$\textbf{Time Complexity} = \sum_{l=0}^{L} C_{\text{sp}}^2 2^l O(k_l)^3 = C_{\text{sp}}^2 \sum_{l=0}^{L} 2^l O(k_l)^3. \tag{53}$$

And the storage for each block is $O(k_l^2)$, with each cluster having C_{sp} blocks. So the memory complexity is

$$\textbf{Memory Complexity} = \sum_{l=0}^{L} C_{\text{sp}} 2^l O(k_l)^2 = C_{\text{sp}} \sum_{l=0}^{L} 2^l O(k_l)^2. \tag{54}$$

Recall k_l is the rank at tree level l. Hence, (53) and (54) show that the overall complexity is a function of rank k_l. Taking into account the rank's growth with electrical size as shown in [5], we can get the time and memory complexity of proposed MMP for different rank scaling. For constant-rank $\mathcal{H}^2$-matrices, since k_l is a constant irrespective of matrix size, the complexity of the proposed direct solution is strictly $O(N)$ in both CPU time and memory consumption, as shown as follows:



(a) (b)

Fig. 6. MMP performance for $\mathbf{A}_{\mathcal{H}^2} \times \mathbf{B}_{\mathcal{H}^2}$ of large-scale capacitance extraction matrices of a 2-layer cross bus structure. (a) Time scaling versus N. (b) Memory scaling versus N.

For constant k_l:

$$\textbf{Time Complexity} = C_{\text{sp}}^2 k_l^3 \sum_{l=0}^{L} 2^l = O(N) \tag{55}$$

$$\textbf{Memory Complexity} = C_{\text{sp}} k_l^2 \sum_{l=0}^{L} 2^l = O(N). \tag{56}$$

For electrodynamic analysis, to ensure a prescribed accuracy, the rank becomes a function of electrical size, and thereby tree level. Different $\mathcal{H}^2$-matrix representations can result in different complexities, because their rank's behavior is different. Using a minimal-rank $\mathcal{H}^2$-representation, as shown by [5], the rank grows linearly with electrical size for general 3-D problems. In a VIE, k_l is proportional to the cubic root of matrix size at level l, because this is the electrical size at level l. Hence for a VIE, (53) and (54) become

For k_l linearly growing with electrical size:

$$\textbf{Time Complexity} = {C_{\text{sp}}}^2 \sum_{l=0}^{L} 2^l \left[\left(\frac{N}{2^l} \right)^{\frac{1}{3}} \right]^3 = O(N \log N) \tag{57}$$

$$\textbf{Memory Complexity} = C_{\text{sp}} \sum_{l=0}^{L} 2^l \left[\left(\frac{N}{2^l} \right)^{\frac{1}{3}} \right]^2 = O(N). \tag{58}$$

So the time complexity of the proposed MMP algorithm for 3-D electrodynamic analysis is $O(NlogN)$, and the memory complexity is $O(N)$.

VII. Numerical Results

In order to demonstrate the accuracy and low computational complexity of the proposed fast $\mathcal{H}^2$-matrix-matrix multiplication for general $\mathcal{H}^2$-matrices, we use $\mathcal{H}^2$-matrices resulting from large-scale capacitance extraction and volume IE (VIE)-based scattering analysis as examples. The capacitance extraction matrix is shown in [7]. The VIE formulation is based on [13], [14], and [16] with SWG vector bases for expanding electric flux density in each tetrahedral element. A variety of large-scale examples involving over one million unknowns are simulated on a single CPU core to examine the accuracy and complexity of the proposed MMP algorithm. The capacitance matrix is used to demonstrate the proposed MMP algorithm performance for constant-rank $\mathcal{H}^2$-matrices. We also simulate large scale 2-D and 3-D scattering examples to examine the time and memory complexity of the proposed MMP for variable rank cases. The computer used has an Intel(R) Xeon(R) CPU E5-2690 v2 running at 3 GHz, and only a *single core* is employed to carry out the computation.

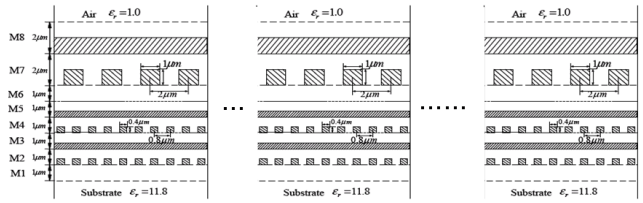


Fig. 7. Illustration of the large-scale M1–M8 on-chip interconnect embedded in inhomogeneous dielectrics [7], with the conductor number increased from 96, 144, 192, 240, 288, to 336.

The accuracy of the proposed MMP is assessed by using the following criterion:

$$\epsilon_{\text{rel}} = \frac{||\mathbf{C}_{\mathcal{H}^2} x - \mathbf{A}_{\mathcal{H}^2}(\mathbf{B}_{\mathcal{H}^2} x)||_F}{||\mathbf{A}_{\mathcal{H}^2}(\mathbf{B}_{\mathcal{H}^2} x)||_F} \tag{59}$$

where $\mathbf{A}_{\mathcal{H}^2} \times (\mathbf{B}_{\mathcal{H}^2} \times x)$ is used as the reference solution, since given an $\mathcal{H}^2$ matrix, an MVP can be carried out without any approximation, as shown in [1]. In generating the reference solution, we first compute $y = \mathbf{B}_{\mathcal{H}^2} \times x$, and then compute $\mathbf{A}_{\mathcal{H}^2} \times y$, both of which are done in exact arithmetic. The proposed solution is generated by first computing an MMP of $\mathbf{A}_{\mathcal{H}^2}\mathbf{B}_{\mathcal{H}^2}$ to obtain $\mathbf{C}_{\mathcal{H}^2}$, and then compute $\mathbf{C}_{\mathcal{H}^2} x$. The x is chosen to be a random vector to assess the accuracy for arbitrary vectors.

A. Two-Layer Cross Bus

The first example is the capacitance extraction of a two-layer cross bus structure, as illustrated in Fig. 5. In each layer,

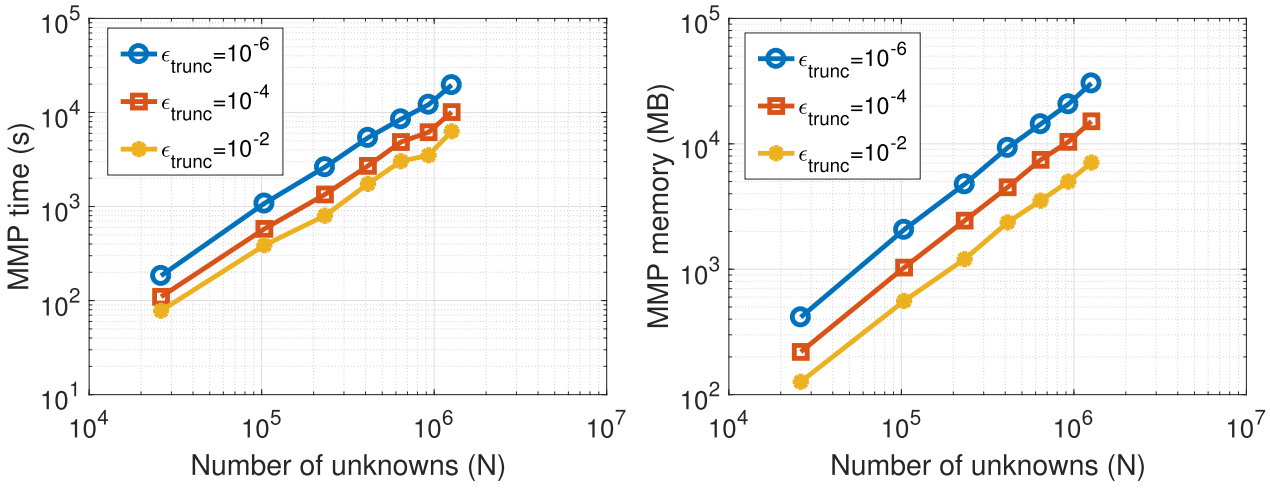


Fig. 8. MMP performance for $\mathbf{A}_{\mathcal{H}^2} \times \mathbf{B}_{\mathcal{H}^2}$ of large-scale M1–M8 capacitance extraction in inhomogeneous dielectrics. (a) Time scaling versus N. (b) Memory scaling versus N.

there are m conductors, and each conductor has a dimension of $1 \times 1 \times (2m+1)$ m^3. We simulate a suite of such structures with 16, 32, 64, 128, and 256 conductors, respectively. The parameters used in the $\mathcal{H}^2$-matrix construction are $leafsize = 30$, admissibility condition [1] $\eta = 1.0$, and $\epsilon_{\mathcal{H}^2} = 10^{-4}$. The $\mathcal{H}^2$-matrix for this example is constructed based on the method described in [16], where $\epsilon_{\mathcal{H}^2}$ is ϵ_{acc} in [16]. For the proposed $\mathcal{H}^2$ MMP, the ϵ_{trunc} is chosen to be 10^{-2}, 10^{-4}, and 10^{-6}, respectively to examine the error controllability. The matrix $\mathbf{A}$ is chosen to be the discretized double layer potential operator, whose ijth element can be written as

$$\mathbf{A}_{ij} = \int\int_{S_i} \int\int_{S_j} \frac{(\mathbf{r}-\mathbf{r}') \cdot \hat{n}(\mathbf{r}')}{4\pi |\mathbf{r}-\mathbf{r}'|^3} dS' dS \tag{60}$$

where $\hat{n}$ denotes a unit vector normal to S_j, $\mathbf{r}$ denotes the position vector of an observation point, and $\mathbf{r}'$ denotes that of a source point. The $\mathbf{B}$ matrix is chosen to be the discretized single layer potential operator, whose ijth element is

$$\mathbf{B}_{ij} = \int\int_{S_i} \int\int_{S_j} \frac{1}{4\pi |\mathbf{r}-\mathbf{r}'|} dS' dS. \tag{61}$$

As shown in Fig. 6, the proposed MMP exhibits clear linear complexities in time and memory regardless of the choice of ϵ_{trunc}. Certainly, the smaller the ϵ_{trunc}, the larger the computational cost. From Table I, we can see the accuracy of the proposed MMP is very good, and it is also controllable. By choosing a smaller ϵ_{trunc}, the MMP error can be reduced.

B. Large-Scale 3-D M1–M8 On-Chip Interconnect in Inhomogeneous Dielectrics

We consider a more complicated example, which is a large-scale on-chip interconnect embedded in multiple layers of dielectrics as shown in Fig. 7. The relative permittivity of the interconnect structure is 3.9 in M1, 2.5 from M2 to M6, and 7.0 from M7 to M8. The structure involves 48 conductors, the discretization of which results in 26 112 unknowns. Let z be the vertical direction from M1 to M8, and y pointing into the article. The 48-conductor structure is duplicated

TABLE I

$\mathcal{H}^2$ MMP ERROR ϵ_{rel} AT DIFFERENT ϵ_{trunc} FOR LARGE-SCALE CAPACITANCE EXTRACTION MATRICES AS A FUNCTION OF N

N	4,480	17,152	67,072	265,216	1,054,720
$\epsilon_{trunc} = 10^{-2}$	4.35E-2	5.72E-2	5.73E-2	5.80E-2	5.97E-2
$\epsilon_{trunc} = 10^{-4}$	3.71E-3	3.72E-3	3.86E-3	3.80E-3	3.67E-3
$\epsilon_{trunc} = 10^{-6}$	2.82E-4	3.28E-4	3.86E-4	4.50E-4	5.66E-4

horizontally (along x-direction), with the length extended as well along the y-direction, resulting in 96, 144, 192, 240, 288, and 336 conductors, the number of unknowns of which are 26 112, 103 680, 232 704, 413 184, 645 120, 928 512, and 1 263 360, respectively.

An IE-based solution for capacitance extraction results in the following dense system of equations:

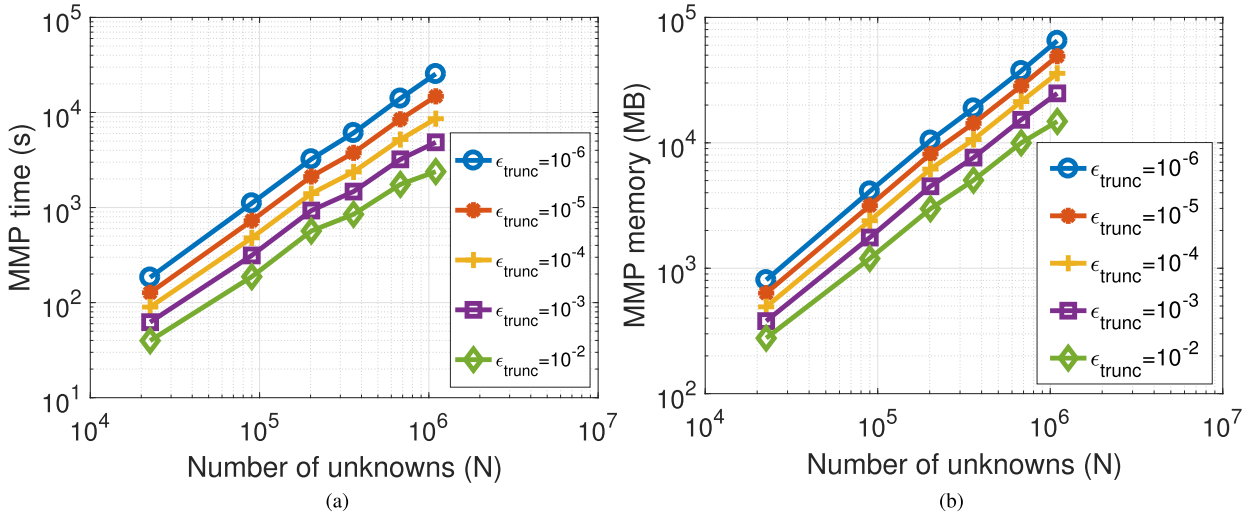
$$\mathbf{G}q = v \tag{62}$$

where

$$\mathbf{G} = \begin{bmatrix} \mathbf{P}_{cc} & \mathbf{P}_{cd} \\ \mathbf{P}_{dc} & \mathbf{P}_{dd} \end{bmatrix}, \quad q = \begin{bmatrix} q_c \\ q_d \end{bmatrix}, \text{ and } v = \begin{bmatrix} v_c \\ 0 \end{bmatrix}$$

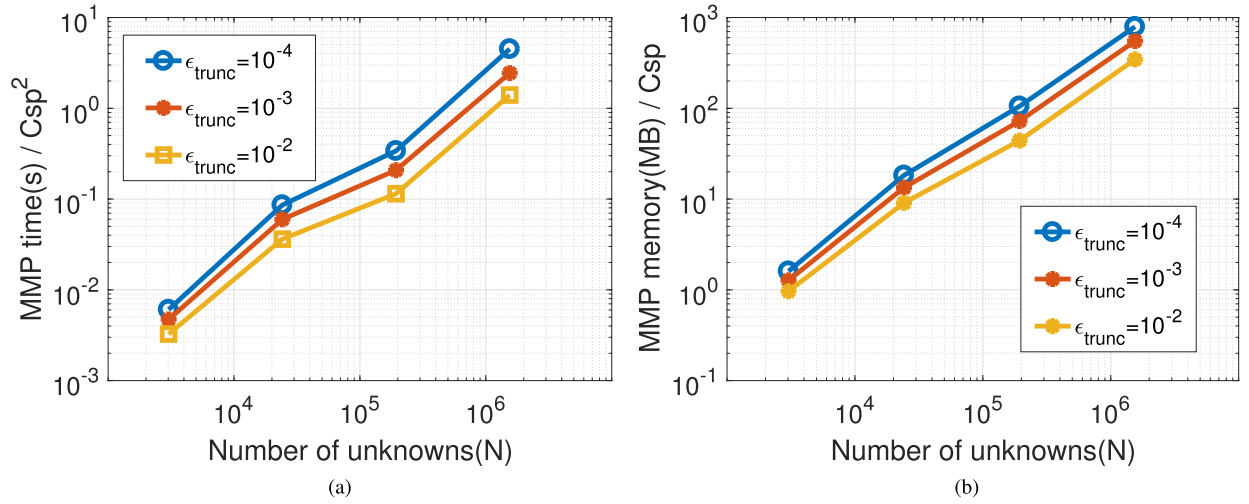
in which q_c and q_d denote charges on the conductor panels, and dielectric-interface panels, respectively. The v_c is the potential attached to a conductor panel. In this example, we choose $\mathbf{A} = \mathbf{B} = \mathbf{G}$, whose matrix entries can be found in [7]. The $\mathcal{H}^2$-tree structure of $\mathbf{G}$ is built using $leafsize = 20$, and $\eta = 4.0$. The method in [7] is used to construct the $\mathcal{H}^2$ matrix with a polynomial order equal to three in every direction. For the proposed $\mathcal{H}^2$-MMP, the ϵ_{trunc} is chosen to be 10^{-2}, 10^{-4}, and 10^{-6}, respectively, to examine the error controllability.

As shown in Fig. 8, the proposed MMP exhibits clear linear complexities in time and memory regardless of the choice of ϵ_{trunc}. In addition, the smaller the ϵ_{trunc}, the larger the computational cost, which is as expected. The accuracy of the proposed MMP is assessed by using the same criterion as shown in (59). From Table II, very good accuracy is observed, and it is also controllable via the choice of ϵ_{trunc}. Smaller ϵ_{trunc} results in better accuracy.



(a) (b)

Fig. 9. MMP performance for $\mathbf{A}_{\mathcal{H}^2} \times \mathbf{A}_{\mathcal{H}^2}$ of 2-D slab scattering from 4λ to 28λ. (a) Time scaling versus N. (b) Memory scaling versus N.



(a) (b)

Fig. 10. MMP performance for $\mathbf{A}_{\mathcal{H}^2} \times \mathbf{A}_{\mathcal{H}^2}$ of scattering from a 3-D cube array. (a) Time scaling versus N. (b) Memory scaling versus N.

TABLE II

$\mathcal{H}^2$ MMP ERROR MEASURED BY ϵ_{rel} SHOWN IN (59) AT DIFFERENT ϵ_{trunc} FOR LARGE-SCALE M1–M8 CAPACITANCE EXTRACTION MATRICES IN INHOMOGENEOUS MATERIALS AS A FUNCTION OF N

N	26,112	103,680	232,704	413,184	645,120	928,512	1,263,360
ϵ_{rel} ($\epsilon_{trunc} = 10^{-2}$)	5.31E-3	4.95E-3	4.68E-3	4.77E-3	4.98E-3	4.72E-3	4.43E-3
ϵ_{rel} ($\epsilon_{trunc} = 10^{-4}$)	3.58E-4	4.06E-4	3.03E-4	4.32E-4	3.65E-4	3.14E-4	3.51E-4
ϵ_{rel} ($\epsilon_{trunc} = 10^{-6}$)	2.55E-5	2.77E-5	3.30E-5	2.94E-5	2.17E-5	3.37E-5	2.31E-5

C. Large-Scale Dielectric Slab Scattering

We then simulate a dielectric slab with $\epsilon_r = 2.54$ at 300 MHz, the structure of which is the same as [16, Fig. 10]. The thickness of the slab is fixed to be $0.1\lambda_0$. The width and length are simultaneously increased from $4\lambda_0$, $8\lambda_0$, $16\lambda_0$, to $28\lambda_0$. With a mesh size of $0.1\lambda_0$, the resultant N ranges from 22 560 to 1 098 720 for this suite of slab structures. The parameters used in the $\mathcal{H}^2$-matrix construction are $leafsize = 40$, $\eta = 2.0$, and $\epsilon_{\mathcal{H}^2} = 10^{-3}$. The $\mathcal{H}^2$-matrix is constructed based on the method described in [16]. For the proposed $\mathcal{H}^2$ MMP, the ϵ_{trunc} is chosen to be 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, and 10^{-6}, respectively, to examine the computational complexity and error controllability of the proposed MMP.

The IE formulation used in this example is the following VIE:

$$\mathbf{E}^i(\mathbf{r}) = \frac{\mathbf{D}(\mathbf{r})}{\epsilon(\mathbf{r})} - \int_V \left[\mu_0\omega^2\kappa(\mathbf{r}')\mathbf{D}(\mathbf{r}') + \nabla\left(\nabla'\cdot\left(\kappa(\mathbf{r})\frac{\mathbf{D}(\mathbf{r}')}{\epsilon_0}\right)\right)\right] g(\mathbf{r},\mathbf{r}')dv' \tag{63}$$

where $g(\mathbf{r},\mathbf{r}') = e^{-jk_0|\mathbf{r}-\mathbf{r}'|}/4\pi|\mathbf{r}-\mathbf{r}'|$, ω being the angular frequency, κ the contrast ratio defined as $(\epsilon(\mathbf{r}) - \epsilon_0)/\epsilon(\mathbf{r})$, $\mathbf{D}(\mathbf{r}')$ the electric flux density, while k_0 is the free space wavenumber. By expanding the unknown electric flux density

TABLE III

$\mathcal{H}^2$ MMP ERROR ϵ_{rel} AS SHOWN IN (59) FOR THE 2-D SLAB SCATTERING PROBLEM FOR DIFFERENT ϵ_{trunc} AS A FUNCTION OF N

N	22,560	89,920	359,040	1,098,720
ϵ_{rel} ($\epsilon_{trunc} = 10^{-2}$)	8.54E-3	1.06E-2	1.49E-2	1.07E-2
ϵ_{rel} ($\epsilon_{trunc} = 10^{-3}$)	2.52E-3	3.17E-3	4.23E-3	3.79E-3
ϵ_{rel} ($\epsilon_{trunc} = 10^{-4}$)	7.86E-4	9.76E-4	1.38E-3	1.23E-3
ϵ_{rel} ($\epsilon_{trunc} = 10^{-5}$)	2.91E-4	3.37E-4	4.22E-4	4.11E-4
ϵ_{rel} ($\epsilon_{trunc} = 10^{-6}$)	8.04E-5	9.85E-5	1.27E-4	1.36E-4

$\mathbf{D}(\mathbf{r}')$ in terms of SWG basis functions each with a coefficient D_n, and then testing the resulting equation using Galerkin method with $\mathbf{D}_m(\mathbf{r})$, we obtain the following linear system of VIE:

$$\mathbf{Z}D = E. \tag{64}$$

A and **B** are both chosen to be equal to **Z** in this example. The detailed expression of **Z** can be found in [16, eq. (3)].

Based on [5], the rank's growth rate with electrical size for 2-D slab is lower than linear, and being a square root of the log-linear of the electric size. Substituting such a rank's growth into the complexity analysis in (53) and (54), we obtain linear complexity in both memory and time. In Fig. 9(a), we plot the MMP time with respect to N, for all different choices of ϵ_{trunc}. It is clear that the smaller ϵ_{trunc} value, the larger the MMP time. However, the complexity remains the same as linear regardless of the choice of ϵ_{trunc}. The memory cost is plotted in Fig. 9(b). Obviously, it scales linearly with the number of unknowns. The error of the proposed MMP is measured in the same way as shown in (59). In Table III, we list the error as a function of ϵ_{trunc}. Excellent accuracy can be observed in the entire unknown range. Furthermore, the accuracy can be controlled by ϵ_{trunc}, and overall smaller ϵ_{trunc} results in better accuracy.

In addition, we compare the CPU run time, memory and accuracy of the proposed MMP with those of [1] using this example. The $\epsilon_{\text{trunc}} = 1e-2$ is chosen in the proposed algorithm so that a similar level of MMP accuracy can be generated for a fair comparison. For the $N = 359\,040$ case, the CPU time of the algorithm in [1] is found to be 875.54 s, whereas the proposed takes 849.08 s only. The memory of [1] is 4.36 GB, while the proposed takes 5.03 GB since it stores additional quantities. As for accuracy, [1] algorithm yields an MMP error of $4.59e-2$; while the proposed is $1.49e-2$. As can be seen, although the proposed algorithm updates cluster bases while [1] does not, the proposed does not cost a longer CPU time. This can be attributed to its one-way tree traversal procedure, as well as the fact that the time spent on updating cluster bases is not much as compared to other computations.

D. Scattering From Large-Scale Array of Dielectric Cubes

Next, we simulate a large-scale array of dielectric cubes at 300 MHz, whose structure is the same as [16, Fig. 13]. The relative permittivity of the cube is $\epsilon_r = 4.0$. Each cube is of size $0.3\lambda_0 \times 0.3\lambda_0 \times 0.3\lambda_0$. The distance between adjacent cubes is kept to be $0.3\lambda_0$. The number of the cubes is increased along the x-, y-, and z- directions simultaneously from 2 to 16, thus producing a 3-D cube array from $2 \times 2 \times 2$ to $16 \times 16 \times 16$ elements. The number of unknowns N is, respectively, 3024, 24 192, 193 536, and 1 548 288 for these arrays. Like previous example, **A** and **B** are both chosen to be equal to VIE-based system matrix **Z**, whose matrix elements can be found in [16, eq. (3)]. During the construction of $\mathcal{H}^2$-matrix, we set $leafsize = 20$, $\eta = 1$, and $\epsilon_{\mathcal{H}^2} = 10^{-2}$. For the proposed $\mathcal{H}^2$ MMP, the ϵ_{trunc} is chosen as 10^{-2}, 10^{-3}, and 10^{-4}.

TABLE IV

C_{sp} AS A FUNCTION OF N FOR THE DIELECTRIC CUBE ARRAY

N	3024	24192	193536	1548288
C_{sp}	16	42	95	126

TABLE V

$\mathcal{H}^2$ MMP ERROR ϵ_{rel} AT DIFFERENT ϵ_{trunc} FOR 3-D CUBE ARRAY

N	3024	24192	193536	1548288
ϵ_{rel} of [1]	9.02E-2	1.01E-1	1.77E-1	2.74E-1
ϵ_{rel} (ϵ_{trunc}=10^{-2})	1.91E-2	2.38E-2	3.82E-2	6.58E-2
ϵ_{rel} (ϵ_{trunc}=10^{-3})	5.51E-3	7.23E-3	1.06E-2	2.16E-2
ϵ_{rel} (ϵ_{trunc}=10^{-4})	1.48E-3	2.46E-3	3.69E-3	8.09E-3

For a cubic growth of unknowns in 3-D problems, we observe that constant C_{sp} is quite different for different unknowns, as can be seen from Table IV. It is thus important to analyze the performances of the proposed MMP as $Memory/C_{\text{sp}}$ and $Multiplication\ time/C_{\text{sp}}^2$ respectively to examine the true scaling rate. In Fig. 10(a) and (b), we plot the $\mathcal{H}^2$-matrix-matrix multiplication time divided by C_{sp}^2, and the storage cost normalized with C_{sp} with respect to N. As can be seen, their scaling rate with N agrees very well with our theoretical complexity analysis. For the largest case, which is a $16 \times 16 \times 16$ cube array having thousands of cube elements, the error is still controlled to be as small as 0.809% using $\epsilon_{\text{trunc}} = 10^{-4}$. The error of the proposed MMP is listed in Table V for this example, which again reveals excellent accuracy and error controllability of the proposed MMP.

We also compare the accuracy of the proposed MMP with existing MMP [1] using this 3-D example. As shown in Table V, the proposed MMP has much better accuracy, and also it is controllable. It is worth mentioning that the H2Lib at http://www.h2lib.org/doc/files.html is not ready for computing the matrix-matrix multiplications done in this article, and hence we implemented our own version of the MMP of [1] to compare with the proposed new MMP algorithm.

VIII. CONCLUSION

In this article, we develop a fast accuracy-controlled algorithm to compute $\mathcal{H}^2$-MMPs for general $\mathcal{H}^2$-matrices. This proposed algorithm not only has *explicitly* controlled accuracy, but also generates a rank-minimized representation of the product matrix based on prescribed accuracy. The row and column cluster bases are instantaneously changed so that the new matrix content generated during the MMP can be accurately represented. This ensures that each multiplication performed in the proposed MMP is well controlled by accu-

racy. Meanwhile, we retain the complexity to be linear for constant-rank $\mathcal{H}^2$-matrices. The proposed algorithm has been applied to calculate $\mathcal{H}^2$-MMPs for large-scale capacitance extraction matrices whose kernel is static and real-valued and electrically large VIEs whose kernel is oscillatory and complex-valued. For constant-rank $\mathcal{H}^2$-matrices, the proposed MMP has an $O(N)$ complexity in both time and memory. For rank growing with the electrical size linearly, the proposed MMP has an $O(NlogN)$ complexity time and $O(N)$ complexity in memory. $\mathcal{H}^2$-matrix products with millions of unknowns are simulated on a single core CPU in fast CPU run time. Comparisons with existing $\mathcal{H}^2$-MMP algorithm have demonstrated clear advantages of the proposed new MMP algorithm.

REFERENCES

[1] S. Börm, "H^2-matrix arithmetics in linear complexity," *Computing*, vol. 77, pp. 1–28, Feb. 2006.

[2] S. Börm, "Efficient numerical methods for non-local operators: H^2-matrix compression, algorithms and analysis," *Eur. Math. Soc. Tracts Math.*, vol. 14, 2006.

[3] H. Liu and D. Jiao, "Existence of H-matrix representations of the inverse finite-element matrix of electrodynamic problems and H-based fast direct finite-element solvers," *IEEE Trans. MTT*, vol. 58, no. 12, pp. 3697–3709, 2010.

[4] B. Zhou and D. Jiao, "Direct finite element solver of linear complexity for large-scale 3-d electromagnetic analysis and circuit extraction," *IEEE Trans. MTT*, vol. 63, no. 10, pp. 3066–3080, Oct. 2015.

[5] W. Chai and D. Jiao, "Theoretical study on the rank of integral operators for broadband electromagnetic modeling from static to electrodynamic frequencies," *IEEE Trans. Compon., Packag., Manuf. Technol.*, vol. 3, no. 12, pp. 2113–2126, Dec. 2013.

[6] M. Li, M. A. Francavilla, D. Ding, R. Chen, and G. Vecchi, "Mixed-form nested approximation for wideband multiscale simulations," *IEEE Trans. Antennas Propag.*, vol. 66, no. 11, pp. 6128–6136, Nov. 2018.

[7] W. Chai and D. Jiao, "Dense matrix inversion of linear complexity for integral-equation-based large-scale 3-D capacitance extraction," *IEEE Trans. Microw. Theory Techn.*, vol. 59, no. 10, pp. 2404–2421, Oct. 2011.

[8] B. Zhou and D. Jiao, "Linear-complexity direct finite element solver accelerated for many right hand sides," in *Proc. IEEE Antennas Propag. Soc. Int. Symp. (APSURSI)*, Jul. 2014.

[9] W. Chai and D. Jiao, "Direct matrix solution of linear complexity for surface integral-equation-based impedance extraction of complicated 3-D structures," *Proc. IEEE*, vol. 101, no. 2, pp. 372–388, Feb. 2013.

[10] S. Börm and K. Reimer, "Efficient arithmetic operations for rank-structured matrices based on hierarchical low-rank updates," *Comput. Vis. Sci.*, vol. 16, no. 6, pp. 247–258, Dec. 2013.

[11] W. C. Chew, J. M. Jin, E. Michielssen, and J. M. Song, *Fast and Efficient Algorithms in Computational Electromagnetics*. Norwood, MA, USA: Artech House, 2001.

[12] M. Ma and D. Jiao, "Accuracy-controlled and rank-minimized H^2-matrix-matrix product with change of cluster bases in linear complexity," in *Proc. IEEE MTT-S Int. Conf. Numer. Electromagn. Multiphys. Modeling Optim. (NEMO)*, May 2019.

[13] D. Schaubert, D. Wilton, and A. Glisson, "A tetrahedral modeling method for electromagnetic scattering by arbitrarily shaped inhomogeneous dielectric bodies," *IEEE Trans. Antennas Propag.*, vol. AP-32, no. 1, pp. 77–85, Jan. 1984.

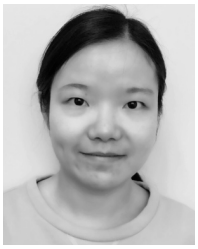
[14] S. Omar and D. Jiao, "O(N) iterative and O(NlogN) direct volume integral equation solvers for large-scale electrodynamic analysis," in *Proc. Int. Conf. Electromagn. Adv. Appl. (ICEAA)*, Aug. 2014, pp. 593–596.

[15] D. Jiao and S. Omar, "Minimal-rank H^2-matrix-based iterative and direct volume integral equation solvers for large-scale scattering analysis," in *Proc. IEEE Int. Symp. Antennas Propag.*, Jul. 2015, pp. 740–741.

[16] S. Omar, M. Ma, and D. Jiao, "Low-complexity direct and iterative volume integral equation solvers with a minimal-rank H^2-representation for large-scale three-dimensional electrodynamic analysis," *IEEE J. Multiscale Multiphys. Comput. Techn.*, vol. 2, pp. 210–223, Oct. 2017.

[17] M. Ma and D. Jiao, "Accuracy controlled H^2-matrix-matrix product in linear complexity and its applications," in *Proc. IEEE Int. Symp. Antennas Propag. USNC/URSI Nat. Radio Sci. Meeting*, Jul. 2018, pp. 2499–2500.

[18] M. Ma and D. Jiao, "Accuracy-controlled and structure-preserved H^2-matrix-matrix product in linear complexity," in *Proc. Int. Conf. Electromagn. Adv. Appl. (ICEAA)*, Sep. 2018.



Miaomiao Ma (Graduate Student Member, IEEE) received the B.S. degree in electronic engineering and information science from the University of Science and Technology of China, Hefei, China, in 2014, and the Ph.D. degree from the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA, in 2019.

Her current research interests include computational electromagnetics, fast and high-capacity numerical methods, and scattering analysis.

Dr. Ma was a recipient of the Cadence's Women in Technology Scholarship in 2018. She also received the Honorable Mention Award from the IEEE International Symposium on Antennas and Propagation in 2016 and 2018, respectively. She was also a recipient of the Best Student Paper Award from the IEEE International Conference on Wireless Information Technology and Systems (ICWITS) and Applied Computational Electromagnetics (ACES) in 2016.

Dan Jiao (Fellow, IEEE) received the Ph.D. degree in electrical engineering from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 2001.

She then worked at the Technology Computer-Aided Design (CAD) Division, Intel Corporation, Santa Clara, CA, USA, until September 2005, as a Senior CAD Engineer, a Staff Engineer, and a Senior Staff Engineer. In September 2005, she joined Purdue University, West Lafayette, IN, USA, as an Assistant Professor with the School of Electrical and Computer Engineering, where she is currently a Professor. She has authored over 300 articles in refereed journals and international conferences. Her current research interests include computational electromagnetics, high-frequency digital, analog, mixed-signal, RF integrated circuit (IC) design and analysis, high-performance VLSI CAD, modeling of microscale and nanoscale circuits, applied electromagnetics, fast and high-capacity numerical methods, fast time-domain analysis, scattering and antenna analysis, RF, microwave, millimeter-wave circuits, wireless communication, and bio-electromagnetics.

Dr. Jiao received the 2013 S. A. Schelkunoff Prize Paper Award from the IEEE Antennas and Propagation Society, which recognizes the best paper published in the IEEE TRANSACTIONS ON ANTENNAS AND PROPAGATION during the previous year. She was among the 21 women faculty selected across the country as the 2014–2015 Fellow of Executive Leadership in Academic Technology and Engineering (ELATE) at Drexel, a national leadership program for women in the academic STEM fields. She has been named a University Faculty Scholar by Purdue University since 2013. She was among the 85 engineers selected throughout the nation for the National Academy of Engineering's 2011 U.S. Frontiers of Engineering Symposium. She was a recipient of the 2010 Ruth and Joel Spira Outstanding Teaching Award, the 2008 National Science Foundation (NSF) CAREER Award, the 2006 Jack and Cathie Kozik Faculty Start-up Award (which recognizes an outstanding new faculty member of the School of Electrical and Computer Engineering, Purdue University), the 2006 Office of Naval Research (ONR) Award under the Young Investigator Program, the 2004 Best Paper Award presented at the Intel Corporation's annual corporate-wide technology conference (Design and Test Technology Conference) for her work on generic broadband model of high-speed circuits, the 2003 Intel Corporation's Logic Technology Development (LTD) Divisional Achievement Award, the Intel Corporation's Technology CAD Divisional Achievement Award, the 2002 Intel Corporation's Components Research the Intel Hero Award, the Intel Corporation's LTD Team Quality Award, and the 2000 Raj Mittra Outstanding Research Award presented by the University of Illinois at Urbana–Champaign. She was the Chair of the Best Paper Awards Committee of the IEEE Antennas and Propagation Society (AP-S) in 2019. She has been the Chair of the IEEE Women in Engineering (WIE) of the IEEE AP-S since 2018. She is the General Chair of the 2019 IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO). She served as a reviewer for many IEEE journals and conferences. She is an Associate Editor of the IEEE JOURNAL ON MULTISCALE AND MULTIPHYSICS COMPUTATIONAL TECHNIQUES.