A Hierarchical Butterfly LU Preconditioner for Two-Dimensional Electromagnetic Scattering Problems Involving Open Surfaces

Yang Liu
Lawrence Berkeley National Laboratory, USA
liuyangzhuan@lbl.gov

Haizhao Yang
Department of Mathematics
National University of Singapore, Singapore
haizhao@nus.edu.sg

February 1, 2019

Abstract

This paper introduces a hierarchical interpolative decomposition butterfly-LU factorization (H-IDBF-LU) preconditioner for solving two-dimensional electric-field integral equations (EFIEs) in electromagnetic scattering problems of perfect electrically conducting objects with open surfaces. H-IDBF-LU leverages the interpolative decomposition butterfly factorization (IDBF) to compress dense blocks of the discretized EFIE operator to expedite its application; this compressed operator also serves as an approximate LU factorization of the EFIE operator leading to an efficient preconditioner in iterative solvers. Both the memory requirement and computational cost of the H-IDBF-LU solver scale as $O(N \log^2 N)$ in one iteration; the total number of iterations required for a reasonably good accuracy scales as $O(1)$ to $O(\log^2 N)$ in all of our numerical tests. The efficacy and accuracy of the proposed preconditioned iterative solver are demonstrated via its application to a broad range of scatterers involving up to 100 million unknowns.

Keywords. Preconditioned iterative solver, interpolative decomposition butterfly factorization, LU factorization, electric-field integral equation (EFIE), scattering.

AMS subject classifications: 44A55, 65R10 and 65T50.

1 Introduction

Iterative and direct surface integral equation (IE) techniques are popular tools for the scattering analysis involving electrically large perfect electrically conducting (PEC) objects. In iterative techniques, fast matvec algorithms including multilevel fast multipole algorithms (MLFMA) [1, 2, 3, 4, 5, 6, 7], directional compression algorithms [8, 9, 10, 11], special function transforms [12, 13, 14, 15, 16], and butterfly factorizations (BFs) [17, 18, 19, 20, 21, 22] can be used to rapidly apply discretized IE operators to trial solution vectors. These techniques typically require $O(KN \log^{\beta} N)$ ($\beta = 1$ or 2) CPU and memory resources, where N is the dimension of the discretized IE operator and K is the number of iterations required for convergence. The widespread adoption and success of fast iterative methods for solving real-world electromagnetic scattering problems can be attributed wholesale to their low computational costs when K is small. However, in many applications when scatterers support resonances or are discretized using multi-scale/dense meshes, the corresponding linear system becomes ill-conditioned and K can be prohibitively large. This motivates a significant amount of research devoted to the design of efficient semi-analytic or analytic preconditioners for integral equations for both open and closed surfaces [23, 24, 25, 25, 26].

Direct solvers do not suffer (to the same degree) from the aforementioned drawbacks, since they construct a compressed representation of the inverse of the discretized IE operator directly. Most state-of-the-art direct methods apply low rank (LR) approximations to compress judiciously selected off-diagonal blocks of the discretized IE operator and its inverse [27, 28, 29, 30, 31, 32, 33]. LR compression schemes are highly efficient and lead to nearly linear scaling direct solvers for electrically small [28, 34], or specially-shaped structures [35, 36, 37, 38]. However, for electrically large and arbitrarily shaped scatterers, the numerical rank of blocks of the discretized IE operators and its inverse is no longer small. As a result, there is no theoretical guarantee of low computational costs for LR schemes in this high-frequency regime; experimentally their CPU and memory requirements have been found to scale as $O(N^\alpha \log^\beta N)$ ($\alpha \in [2.0, 3.0]$, $\beta \geq 1$) and $O(N^\alpha \log N)$ ($\alpha \in [1.3, 2.0]$), respectively. More recently, butterfly factorizations have been applied to construct reduced-complexity direct solvers in the high-frequency regime [39, 40, 41]. These solvers construct butterfly factorizations with constant ranks for blocks (that are LR less compressible) in the discretized IE operator and its inverse and rely on fast randomized algorithms to construct the inverse in $O(N^{1.5} \log N)$ operations.

The lack of quasi-linear complexity direct solvers in the high-frequency regime motivates this work to develop a hierarchical butterfly-compressed algebraic preconditioner for solving EFIEs with $O(N \log^2 N)$ CPU and memory complexity per iteration and up to $O(\log^2 N)$ total iterations, for analyzing scattering from electrically large two-dimensional PEC objects with open surfaces. First, the interpolative decomposition butterfly factorization (IDBF) algorithm [20] is used to compress off-diagonal blocks of the discretized IE operator, leading to $O(N \log^2 N)$ construction and application algorithms with the hierarchical IDBF (H-IDBF) of the IE operator. Second, we construct an approximate butterfly-compressed LU factorization of H-IDBF inspired by the work in [40]. In contrast to [40] that computes the LU factorization via randomized butterfly algebra, here the lower and upper triangular parts of the H-IDBF can directly serve as its approximate LU factorization. This is justified by the observation that the discretized IE operator and its LU factors exhibit similar oscillation patterns after a proper row-wise/column-wise ordering. This approximate LU factorization permits construction of a quasi-linear complexity preconditioner for EFIEs when applied to a wide range of scatterers. Compared to the analytic preconditioners in [25, 26], the proposed H-IDBF-LU preconditioner is capable of analyzing scattering from electrically large objects involving up to 100 million unknowns.

2 Interpolative Decomposition Butterfly Factorization (IDBF)

2.1 Overview

Since the IDBF will be applied repeatedly in this paper, we briefly revisit the $O(N \log N)$ IDBF algorithm proposed in [20] for a complementary low-rank matrix $K \in \mathbb{C}^{M \times N}$ with $M \approx N$. Let X and Ω be the row and column index sets of K. Two trees T_X and T_Ω of the same depth $L = O(\log N)$, associated with X and Ω respectively, are constructed by dyadic partitioning with approximately equal node sizes with leaf node sizes no larger than n_0. Denote the root level of the tree as level 0 and the leaf one as level L. Such a matrix K of size $M \times N$ is said to satisfy the **complementary low-rank property** if for any level ℓ, any node A in T_X at level ℓ, and any node B in T_Ω at level $L - \ell$, the submatrix $K_{A,B}$, obtained by restricting K to the rows indexed by the points in A and the columns indexed by the points in B, is numerically low-rank. See Figure 1 for an illustration of low-rank submatrices in a complementary low-rank matrix of size $16n_0 \times 16n_0$.

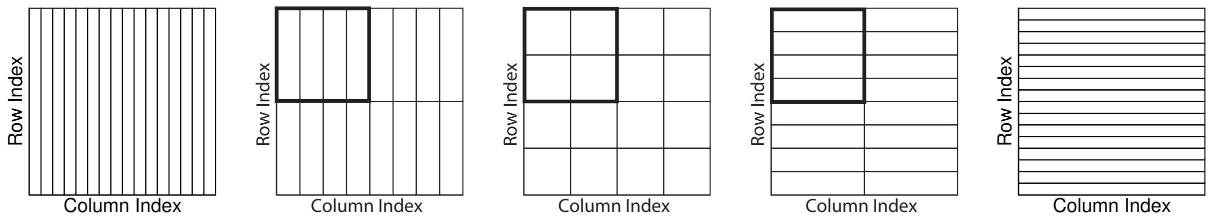

Figure 1: Hierarchical decomposition of the row and column indices of a complementary low-rank matrix of size $16n_0 \times 16n_0$. The trees T_X (T_Ω) has a root containing $16n_0$ column (row) indices and leaves containing n_0 row (column) indices. The rectangles above indicate the low-rank submatrices that will be factorized in IDBF.

Given K, or equivalently an $O(1)$ algorithm to evaluate an arbitrary entry of K, IDBF aims at constructing a data-sparse representation of K using the ID of low-rank submatrices in the complementary low-rank structure (see Figure 1) in the following form:

$$K \approx U^L U^{L-1} \cdots U^h S^h V^h \cdots V^{L-1} V^L, \tag{1}$$

where the depth $L = O(\log N)$ is assumed to be even without loss of generality, $h = L/2$ is a middle level index, and all factors are sparse matrices with $O(N)$ nonzero entries. Storing and applying IDBF requires only $O(N \log N)$ memory and time.

In what follows, uppercase letters will generally denote matrices, while the lowercase letters c, p, q, r, and s denote ordered sets of indices. For a given index set c, its cardinality is written as $|c|$. Given a matrix A, A_{pq}, $A_{p,q}$, or $A(p,q)$ is the submatrix with rows and columns restricted to the index sets p and q, respectively. We also use the notation $A_{:,q}$ to denote the submatrix with columns restricted to q. $s:t$ is an index set containing indices $\{s, s+1, s+2, \ldots, t-1, t\}$. For the sake of simplicity, we assume that $N = 2^L n_0$, where $n_0 = O(1)$ is the number of column or row indices in a leaf in the dyadic trees of row and column spaces, i.e., T_X and T_Ω, respectively. In practical numerical implementations, this assumption is not required.

2.2 Linear scaling Interpolative Decompositions

Suppose A has rank k, a rank revealing QR decomposition to $A_{s,:}$ gives

$$A_{s,:}\Lambda = QR = Q[R_1 \; R_2], \tag{2}$$

where s is an index set containing tk carefully selected rows of A with t as an oversampling parameter, $Q \in \mathbb{C}^{tk \times k}$ is an orthogonal matrix, $R \in \mathbb{C}^{k \times n}$ is upper trapezoidal, and $\Lambda \in \mathbb{C}^{n \times n}$ is a carefully chosen permutation matrix such that $R_1 \in \mathbb{C}^{k \times k}$ is nonsingular. These tk rows can be chosen from the Mock-Chebyshev grids of the row indices as in [22, 42, 43]. Let

$$A_{s,q} = QR_1, \quad T = R_1^{-1} R_2, \tag{3}$$

then

$$A_{:,p} \approx A_{:,q} T, \tag{4}$$

which is equivalent to the traditional form of a column ID,

$$A \approx A_{:,q}[I\ T]\Lambda^* := A_{:,q}V, \tag{5}$$

where q is the complementary set of p, * denotes the conjugate transpose of a matrix, and V is the *column interpolation matrix.*. It can be easily shown that all the steps above require only $O(k^2n)$ operations and $O(kn)$ memory.

In practice, the true rank of A is not available i.e., k is unknown. As is in standard randomized algorithms, we could choose to fix a test rank $k \le n$ or fix the approximation accuracy ϵ and find a numerical rank k_ϵ such that

$$\|A - A_{:,q}V\|_2 \le O(\epsilon) \tag{6}$$

with $T \in \mathbb{C}^{k_\epsilon \times (n-k_\epsilon)}$ and $V \in \mathbb{C}^{k_\epsilon \times n}$. We refer to this linear scaling column ID with an accuracy tolerance ϵ and a rank parameter k as (ϵ, k)-*cID* ((ϵ, k)-*cID* for short). For convenience, we will drop the term (ϵ, k) when it is not necessary to specify it.

Similarly, a row ID for the matrix $A \in \mathbb{C}^{m\times n}$

$$A \approx \Lambda[I\ T]^* A_{q,:} := UA_{q,:} \tag{7}$$

can be attained by performing *cID* on A^* with $O(k^2m)$ operations and $O(km)$ memory. We refer to this linear scaling row ID as (ϵ, k)-*rID* and U as the *row interpolation matrix.*

2.3 Leaf-root complementary skeletonization (LRCS)

Assume that at the leaf level of the row (and column) dyadic trees, the row index set r (and the column index set c) of A are divided into leaves $\{r_i\}_{1\le i\le m}$ (and $\{c_i\}_{1\le i\le m}$) in the following way:

$$r = [r_1, r_2, \cdots, r_m] \qquad (\text{and } c = [c_1, c_2, \cdots, c_m]), \tag{8}$$

with $|r_i| = n_0$ (and $|c_i| = n_0$) for all $1 \le i \le m$, where $m = 2^L = \frac{N}{n_0}$, $L = \log_2 N - \log_2 n_0$, and $L+1$ is the depth of the dyadic trees T_X (and T_Ω). *rID* is applied to each $A_{r_i,:}$ to compute the row interpolation matrix and denote it as U_i; the associated skeleton indices is denoted as $\hat{r}_i \subset r_i$. Let $\hat{r} = [\hat{r}_1, \hat{r}_2, \cdots, \hat{r}_m]$, then $A_{\hat{r},:}$ is the important skeleton of A and we can arrange all the small ID factors into a larger matrix factorization as follows:

$$A \approx \begin{pmatrix} U_1 & & & \\ & U_2 & & \\ & & \ddots & \\ & & & U_m \end{pmatrix} \begin{pmatrix} A_{\hat{r}_1,c_1} & A_{\hat{r}_1,c_2} & \dots & A_{\hat{r}_1,c_m} \\ A_{\hat{r}_2,c_1} & A_{\hat{r}_2,c_2} & \dots & A_{\hat{r}_2,c_m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{\hat{r}_m,c_1} & A_{\hat{r}_m,c_2} & \dots & A_{\hat{r}_m,c_m} \end{pmatrix} := UM.$$

Similarly, *cID* is applied to each $A_{\hat{r},c_j}$ to obtain the column interpolation matrix V_j and the skeleton indices $\hat{c}_j \subset c_j$. Then finally we form the LRCS of A as

$$A \approx \begin{pmatrix} U_1 & & & \\ & U_2 & & \\ & & \ddots & \\ & & & U_m \end{pmatrix} \begin{pmatrix} A_{\hat{r}_1,\hat{c}_1} & A_{\hat{r}_1,\hat{c}_2} & \dots & A_{\hat{r}_1,\hat{c}_m} \\ A_{\hat{r}_2,\hat{c}_1} & A_{\hat{r}_2,\hat{c}_2} & \dots & A_{\hat{r}_2,\hat{c}_m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{\hat{r}_m,\hat{c}_1} & A_{\hat{r}_m,\hat{c}_2} & \dots & A_{\hat{r}_m,\hat{c}_m} \end{pmatrix} \begin{pmatrix} V_1 & & & \\ & V_2 & & \\ & & \ddots & \\ & & & V_m \end{pmatrix} := USV. \tag{9}$$

For a concrete example, Figure 2 visualizes the non-zero pattern of the LRCS in (9).

It is noteworthy that we only generate and store the skeleton of row and column index sets corresponding to M and S, instead of computing M and S explicitly. Hence, it only takes $O(\frac{k^3}{n_0}N)$ operations and $O(\frac{k^2}{n_0}N)$ memory to generate and store the factorization in (9), since there are $2m = \frac{2N}{n_0}$ IDs in total.

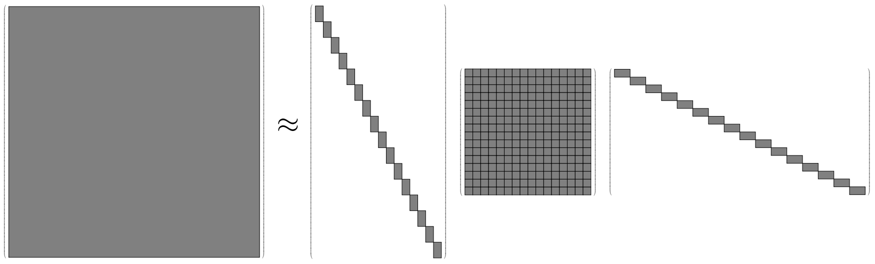

Figure 2: An example of the LRCS in (9) of the complementary low-rank matrix A. Non-zero submatrices in (9) are shown in gray areas.

2.4 Matrix splitting with complementary skeletonization (MSCS)

A complementary low-rank matrix A (with row and column dyadic trees T_X and T_Ω of depth $L+1$ and with $m = 2^L$ leaves) can be split into a 2×2 block matrix, $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$, according to the nodes of the second level of the dyadic trees T_X and T_Ω (right next to the root). As a result, each A_{ij} is also a complementary low-rank matrix with its row and column dyadic trees of $L-1$ levels. For example, A in Figure 1 is a five-level complementary low-rank matrix and A_{11} is a three-level complementary low-rank matrix (see the highlighted submatrices in Figure 1).

Suppose $A_{ij} \approx U_{ij} S_{ij} V_{ij}$, for $i,j = 1,2$, is the LRCS of A_{ij}. Then $A \approx USV$, where

$$U = \begin{pmatrix} U_{11} & & U_{12} & \\ & U_{21} & & U_{22} \end{pmatrix}, \quad S = \begin{pmatrix} S_{11} & & & \\ & & S_{21} & \\ & S_{12} & & \\ & & & S_{22} \end{pmatrix}, \quad V = \begin{pmatrix} V_{11} & \\ & V_{12} \\ V_{21} & \\ & V_{22} \end{pmatrix}. \tag{10}$$

The factorization in (10) is referred as the MSCS. Recall that the middle factor S is not explicitly computed, resulting in a linear scaling algorithm for forming (10). Figure 3 visualizes the MSCS of a complementary low-rank matrix A.

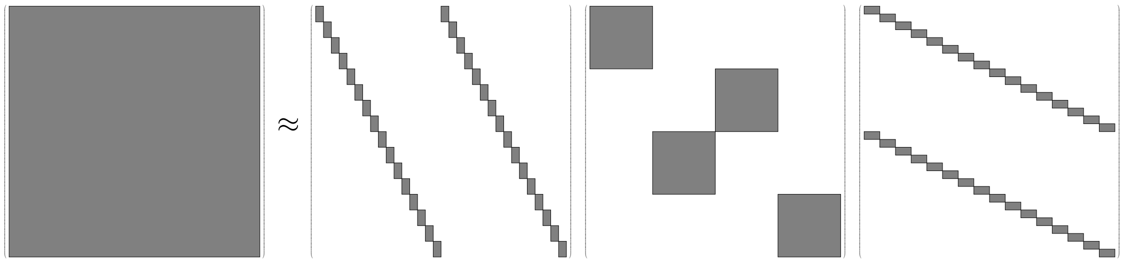

Figure 3: The visualization of a MSCS of a complementary low-rank matrix $A \approx USV$. Non-zero blocks in (10) are shown in gray areas.

2.5 Recursive MSCS

Now we apply MSCS recursively to get the full IDBF. Suppose we have the first level of MSCS as $A \approx U^L S^L V^L$ with

$$U^L = \begin{pmatrix} U_{11}^L & & U_{12}^L & \\ & U_{21}^L & & U_{22}^L \end{pmatrix}, \quad S^L = \begin{pmatrix} S_{11}^L & & & \\ & & S_{21}^L & \\ & S_{12}^L & & \\ & & & S_{22}^L \end{pmatrix}, \quad V^L = \begin{pmatrix} V_{11}^L & \\ & V_{12}^L \\ V_{21}^L & \\ & V_{22}^L \end{pmatrix}. \tag{11}$$

By construction, we know S_{ij}^L are complementary low-rank. Next, apply MSCS to each S_{ij}^L:

$$S_{ij}^L \approx U_{ij}^{L-1} S_{ij}^{L-1} V_{ij}^{L-1}, \tag{12}$$

where

$$\begin{aligned} U_{ij}^{L-1} &= \begin{pmatrix} (U_{ij}^{L-1})_{11} & & (U_{ij}^{L-1})_{12} & \\ & (U_{ij}^{L-1})_{21} & & (U_{ij}^{L-1})_{22} \end{pmatrix}, \\ S_{ij}^{L-1} &= \begin{pmatrix} (S_{ij}^{L-1})_{11} & & & \\ & & (S_{ij}^{L-1})_{21} & \\ & (S_{ij}^{L-1})_{12} & & \\ & & & (S_{ij}^{L-1})_{22} \end{pmatrix}, \\ V_{ij}^{L-1} &= \begin{pmatrix} (V_{ij}^{L-1})_{11} & \\ & (V_{ij}^{L-1})_{12} \\ (V_{ij}^{L-1})_{21} & \\ & (V_{ij}^{L-1})_{22} \end{pmatrix}. \end{aligned} \tag{13}$$

Finally, organizing (12) forms $S^L \approx U^{L-1} S^{L-1} V^{L-1}$ (see its visualization in Figure 4), where

$$U^{L-1} = \begin{pmatrix} U_{11}^{L-1} & & & \\ & U_{21}^{L-1} & & \\ & & U_{12}^{L-1} & \\ & & & U_{11}^{L-1} \end{pmatrix}, \quad S^{L-1} = \begin{pmatrix} S_{11}^{L-1} & & & \\ & & S_{21}^{L-1} & \\ & S_{12}^{L-1} & & \\ & & & S_{22}^{L-1} \end{pmatrix}, \tag{14}$$

$$V^{L-1} = \begin{pmatrix} V_{11}^{L-1} & & & \\ & V_{12}^{L-1} & & \\ & & V_{21}^{L-1} & \\ & & & S_{22}^{L-1} \end{pmatrix},$$

leading to a second level factorization of A as $A \approx U^L U^{L-1} S^{L-1} V^{L-1} V^L$.

Similarly, we can apply MSCS recursively to each S^ℓ and assemble matrix factors hierarchically for $\ell = L, L-1, \ldots, L/2$ to obtain

$$A \approx U^L U^{L-1} \cdots U^h S^h V^h \cdots V^{L-1} V^L, \tag{15}$$

where $h = L/2$. In the entire computation procedure, linear IDs only require $O(1)$ operations for each low-rank submatrix, and hence at most $O(N)$ for each level of factorization, and $O(N \log N)$ for the whole IDBF.

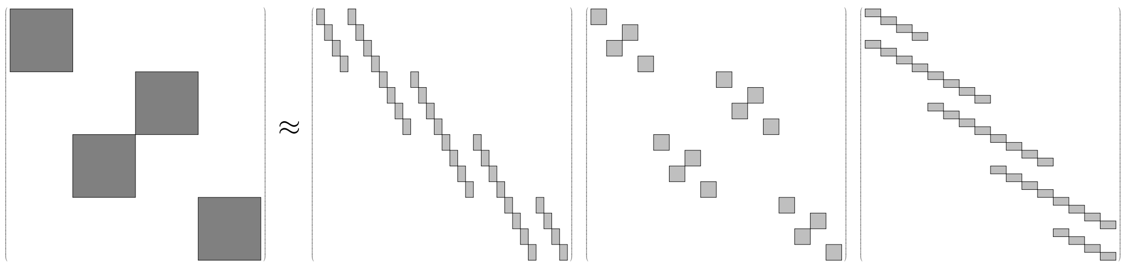

Figure 4: The visualization of the recursive MSCS of $S^L = U^{L-1}S^{L-1}V^{L-1}$.

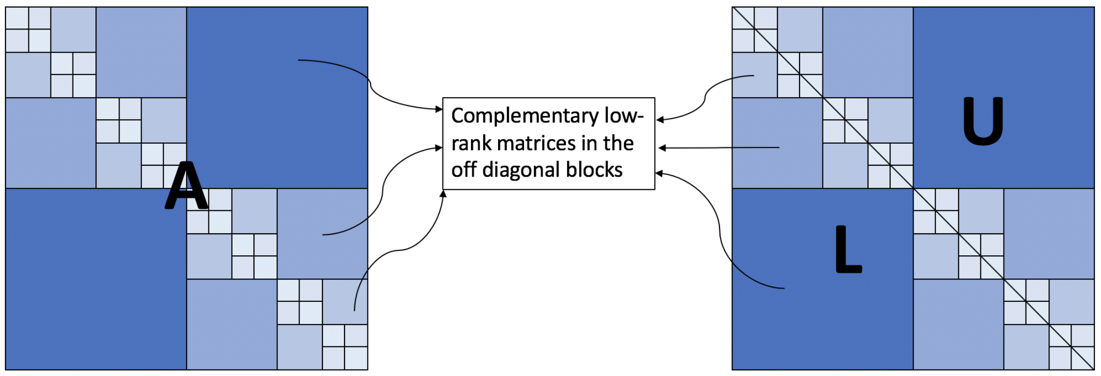

Figure 5: Left: hierarchical decomposition of the impedance matrix A with off-diagonal blocks as complementary low-rank matrices. Right: A is separated into a lower triangular part (denoted as L) with diagonal entries being 1, and a upper triangular part (denoted as U) with diagonal entries equal to those of A.

3 H-IDBF-LU

3.1 Overview of the H-IDBF-LU preconditioner

Here we briefly describe the motivation and workflow of H-IDBF-LU in the context of EFIE for 2D TM$_z$ scattering. Let S denote a PEC cylindrical shell residing in free space. A current $J_z(\rho)$ is induced on S by an incident electric field $E_z^{inc}(\rho)$. Enforcing a vanishing total electric field on S results in the following EFIE:

$$E_z^{inc}(\rho) = \frac{k\eta_0}{4}\int_S J_z(\rho')H_0^{(2)}(k|\rho-\rho'|)ds, \quad \forall \rho \in S.$$

Here, $k = 2\pi/\lambda_0$ is the wavenumber, λ_0 represents the free-space wavelength, η_0 is the intrinsic impedance of free space, and $H_0^{(2)}$ denotes the zeroth-order Hankel function of the second kind. Upon discretization of the current with pulse basis functions and point matching the above EFIE, the following system of linear equations is attained [17]

$$Ax = b, \tag{16}$$

where A is the impedance matrix with

$$A_{ij} = \begin{cases} \frac{k\eta_0 w_j}{4} H_0^{(2)}(k|\rho_i - \rho_j|), & \text{if } i \neq j, \\ \frac{k\eta_0 w_i}{4}\left[1 - \mathrm{i}\frac{2}{\pi}\ln\left(\frac{\gamma k w_i}{4e}\right)\right], & \text{otherwise,} \end{cases}$$

$b_i = E_z^{inc}(\rho_i)$ and $x_i = J_z(\rho_i)$ where $e \approx 2.718$, $\gamma \approx 1.781$, w_i and ρ_i are the length and center of the line segment associated with i-th pulse basis function. In the proposed solver, the linear system in (16) is rescaled to $aAx = ab$, where we choose $a := \frac{1}{\max_i\{|A(i,1)|\}}$ or $a := \|A\|_2$. Here $\|A\|_2$ can be rapidly computed via randomized SVD algorithms. Rescaling is a key step for the proposed LU preconditioner to avoid arithmetic overflow in solution vectors as the iteration count increases. With the abuse of notations, we will still use $Ax = b$ to denote the linear system after rescaling.

It was validated in [17, 39] that A admits a hierarchical complementary low-rank property, i.e., off-diagonal blocks (only well-separated ones if necessary) are complementary low-rank matrices (See Figure 5 (left) for an example). [17] originally proposed a hierarchical compression technique with $O(N \log^2 N)$ operations to compress the impedance matrix A; the IDBF in [20] proposed a more stable algorithm with the same accuracy to compress A. After compression, it requires $O(N \log^2 N)$ operations to apply the impedance matrix and makes it possible to design efficient linear solvers for (16). The algorithm in [20] for compression of the impedance matrix is referred to as hierarchical IDBF (H-IDBF) in this paper.

A theoretical justification of the hierarchical complementary low-rank property can be derived from the fact that the Hankel function admits a smooth amplitude function $\alpha(x)$ and a smooth phase function $\phi(x)$ [44] such that

$$H_0^{(2)}(x) = \alpha(x) e^{i\phi(x)}.$$

Therefore, in the off-diagonal part of A, the submatrix is the discretization of an oscillatory kernel function of an integral operator with a smooth amplitude function and a smooth phase function, which is a complementary low-rank matrix as proved in [45].

Since the H-IDBF of the impedance matrix A leads to an $O(N \log^2 N)$ fast matvec algorithm in a purely algebraic fashion, an immediate question is how to construct a direct solver or an efficient preconditioner for the linear system in (16). It was observed (without proof) in [40] that the L and U factors in the LU factorization of A are also hierarchically complementary low-rank matrices. Hence, [40] proposed an $O(N^{1.5} \log N)$ factorization algorithm based on randomized butterfly algebra to construct hierarchical compressions of the L and U factors, followed by an $O(N \log^2 N)$ block triangular solution algorithm for rapidly application of the matrix inverse to a vector. The lack of a quasi-linear LU factorization algorithm as a direct solver in [40] motivates this work to construct an approximate LU factorization of A in $O(N \log^2 N)$ operations as an efficient preconditioner.

The main observation of the proposed approximate LU factorization of A is that, the lower and upper triangular parts of A have similarly oscillation patterns as the L and U factors of its LU factorization, when the scatterer consists of open surfaces that do not support high-Q resonance. It is also critical that the rows and columns of A are properly ordered such that A exhibits only $O(1)$ discontinuities in each row and column. Figure 6 and 7 show two examples of such scatterers; one is a spiral scatterer and the other one consists of two parallel strips. To be precise, define triangular matrices $\tilde{L}$ and $\tilde{U}$ as follows:

$$\tilde{L}(i,j) = \begin{cases} 0, & \text{if } i < j, \\ 1, & \text{if } i = j, \\ A(i,j), & \text{otherwise,} \end{cases}$$

and

$$\tilde{U}(i,j) = \begin{cases} 0, & \text{if } i > j, \\ A(i,j), & \text{otherwise.} \end{cases}$$

Let L and U be the lower and upper triangular matrices of the LU factorization of A, we observe $L \approx \tilde{L}$, $U \approx \tilde{U}$, and $\tilde{L}^{-1}A\tilde{U}^{-1} \approx I$ numerically, where I is an identity matrix. See Figure 6 and 7 for the visualization of the approximation $\tilde{L}^{-1}A\tilde{U}^{-1} \approx I$ and the eigenvalue distribution of $\tilde{L}^{-1}A\tilde{U}^{-1} \approx I$. Take a spiral scatterer for example, the real (and imaginary) parts of A (Figure 6(d)) and its LU factors (Figure 6(e)) exhibit very similar oscillation patterns. Before preconditioning, there are eigenvalues of A clustered at the origin (Figure 6(b)), while the preconditioned system $\tilde{L}^{-1}A\tilde{U}^{-1}$ has no eigenvalues near the origin (Figure 6(c)). In fact, $\tilde{L}^{-1}A\tilde{U}^{-1}$ exhibits vanishingly small off-diagonal values (Figure 6(f)). The H-IDBFs of $\tilde{L}$ and $\tilde{U}$ are referred to as the H-IDBF-LU of the impedance matrix A in this paper.

Note that $\tilde{L}$ and $\tilde{U}$ are the lower and upper triangular parts of A, whose H-IDBFs are readily available from the H-IDBF of A without any extra cost. Following the block triangular solve algorithm for H-matrices [46], it only requires $O(N\log^2 N)$ operations to apply $\tilde{L}^{-1}$ and $\tilde{U}^{-1}$ to a vector. Hence, the approximate LU factorization results in an $O(N\log^2 N)$ preconditioner for the linear system in (16). To be more specific, instead of directly solving (16) using traditional iterative solvers, we solve a preconditioned linear system

$$\tilde{L}^{-1}A\tilde{U}^{-1}y = \tilde{L}^{-1}b, \tag{17}$$

and

$$\tilde{U}x = y. \tag{18}$$

Since the eigenvalues of $\tilde{L}^{-1}A\tilde{U}^{-1}$ are well separated from 0 and gathered around 1 (see Figure 6 and 7 for two examples), the condition number of $\tilde{L}^{-1}A\tilde{U}^{-1}$ is well controlled and the number of iterations for solving (17) is small. In all of our examples, the iteration number is $O(1)$, $O(\log N)$, or $O(\log^2 N)$ for scatterers consisting of open surfaces. As the construction, application and triangular solve algorithms all require $O(N\log^2 N)$ operations, the overall preconditioner is also quasi-linear.

The remaining task in this section is to introduce the fast construction, application and solution phases of the H-IDBF-LU solver.

3.2 Construction and application of H-IDBF

As we have seen in the previous subsection, the construction of the proposed preconditioner, i.e., the H-IDBFs of $\tilde{L}$ and $\tilde{U}$, is an immediate result of the H-IDBF of A. Hence, it is sufficient to introduce the construction and application of the H-IDBF of A. In fact, these resemble standard techniques for hierarchical matrices [47, 48, 49], and can be summarized in Algorithm 1 and 2, respectively. For illustration purpose, we assume N is a power of 2 and omit the parameters of IDBF in all algorithms. Furthermore, we assume all off-diagonal blocks of A are butterfly compressible (i.e., weak admissibility). However, the proposed solver trivially extends to arbitrary problem sizes and strong admissibility. Let n_0 denote the predefined leaf-level size parameter for all IDBFs, algorithm 1 constructs a IDBF of $L = \log_2 N - \log_2 n_0 - v$ levels for each off-diagonal block of dimension $N/2^v$. Once constructed, algorithm 2 can apply the compressed A to arbitrary vectors by accumulating the matvec results of each submatrix.

```
Data: A hierarchical complementary low-rank matrix $A \in \mathbb{C}^{N\times N}$ or an algorithm to
      evaluate an arbitrary entry of $A$ in $O(1)$ operations.
Result: The H-IDBF of $A$, denoted as $F$, stored in a data structure consisting of four parts:
        $F_{11}$, $F_{12}$, $F_{21}$, and $F_{22}$.
1 if $N \le n_0$ then
2     for $i = 1, 2$ and $j = 1, 2$ do
3         Let $F_{ij} = A((1 : N/2) + (i-1)N/2, (1 : N/2) + (j-1)N/2)$;
4 else
5     Recursively apply Algorithm 1 with $A(1 : N/2, 1 : N/2)$ as the input to obtain $F_{11}$ as the
      corresponding output;
6     Recursively apply Algorithm 1 with $A(N/2 + 1 : N, N/2 + 1 : N)$ as the input to obtain
      $F_{22}$ as the corresponding output;
7     Construct the IDBF of $A(1 : N/2, N/2 + 1 : N)$ and store it in $F_{12}$;
8     Construct the IDBF of $A(N/2 + 1 : N, 1 : N/2)$ and store it in $F_{21}$;
```

Algorithm 1: An $O(N \log^2 N)$ recursive algorithm for constructing the H-IDBF of a hierarchical complementary low-rank matrix A.

```
Data: A H-IDBF of $A \in \mathbb{C}^{N\times N}$ and vectors $v, u \in \mathbb{C}^N$.
Result: $u$ += $Av$.
1 if $N \le n_0$ then
2     for $i = 1, 2$ do
3         $u((1 : N/2) + (i-1)N/2)$ += $F_{i1}u(1 : N/2) + F_{i2}u(N/2 + 1 : N)$;
4 else
5     Recursively apply Algorithm 2 to compute $u(1 : N/2)$ += $F_{11}v(1 : N/2)$;
6     Recursively apply Algorithm 2 to compute $u(N/2 + 1 : N)$ += $F_{22}v(N/2 + 1 : N)$;
7     Apply the IDBF stored as $F_{21}$ to get $u(N/2 + 1 : N)$ += $F_{21}v(1 : N/2)$;
8     Apply the IDBF stored as $F_{12}$ to get $u(1 : N/2)$ += $F_{12}v(N/2 + 1 : N)$;
```

Algorithm 2: An $O(N \log^2 N)$ recursive algorithm for applying the H-IDBF of a hierarchical complementary low-rank matrix A to a vector v.

3.3 Solution of H-IDBF-LU

The previous subsection has introduced $O(N \log^2 N)$ algorithms for constructing and applying the H-IDBF of A and forming $\tilde{L}$ and $\tilde{U}$ as a preconditioner. Next, we will introduce $O(N \log^2 N)$ algorithms for applying the inverse of $\tilde{L}$ and $\tilde{U}$ to a vector in this subsection, i.e, solving triangular systems in a format of H-IDBFs. Similarly to solving a triangular system in a format of H-matrix, the triangular solver for H-IDBF can also be done recursively as in Algorithm 3 and 4.

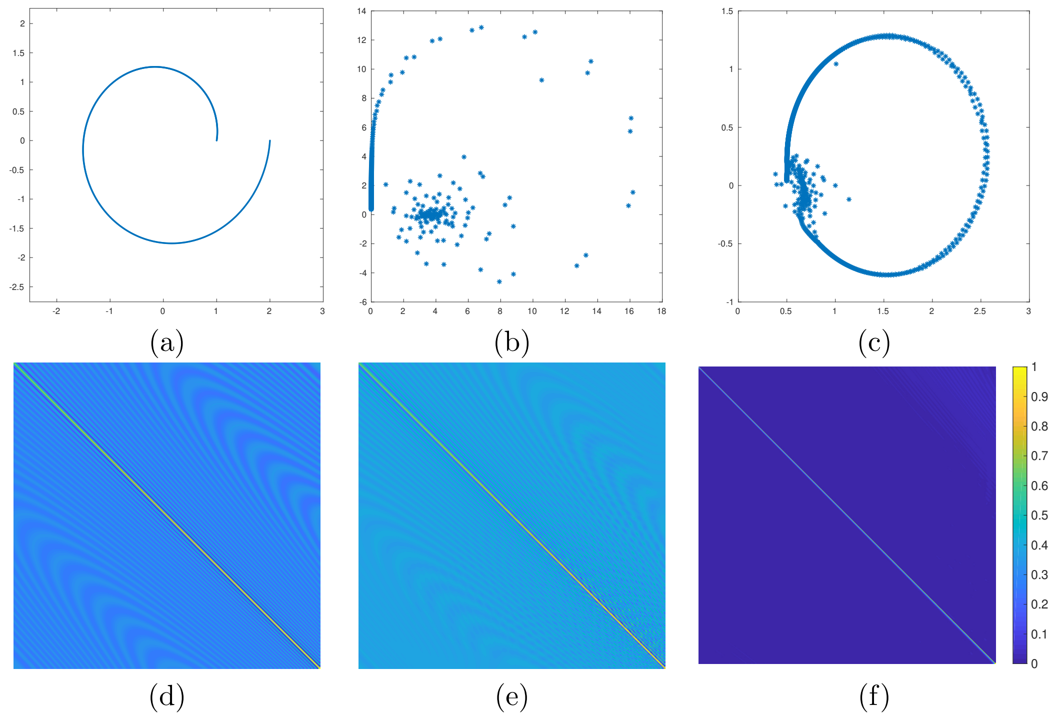

Figure 6: (a) A spiral scatterer. (b) The eigenvalue distribution of the impedance matrix A corresponding to (a). (c) The eigenvalue distribution of $\tilde{L}^{-1}A\tilde{U}^{-1}$, the impedance matrix after preconditioning. (d) The real part of A. (e) The real part of $L+U-I$, where L and U are the LU factors of A, and I is an identity matrix. (f) The magnitude of the entries of $\tilde{L}^{-1}A\tilde{U}^{-1}$.

```
  Data: A H-IDBF of $L \in \mathbb{C}^{N\times N}$ and a vector $b \in \mathbb{C}^N$.
  Result: A vector $x \in \mathbb{C}^N$ equal to $L^{-1}b$.
1 if $N \le n_0$ then
2     Solve the triangualr system $Lx = b$ via standard LAPACK;
3 else
4     Partition the lower triangular system as
          $$L = \begin{pmatrix} F_{11} & 0 \\ F_{21} & F_{22} \end{pmatrix};$$
      Recursively apply Algorithm 3 with $F_{11}$ and $b(1:N/2)$ as the input to obtain
      $x(1:N/2)$ as the corresponding output;
5     Apply Algorithm 2 with $-F_{21}$ to obtain $b(N/2+1:N) -= F_{21}x(1:N/2)$;
6     Recursively apply Algorithm 3 with $F_{22}$ and $b(N/2+1:N)$ as the input to obtain
      $x(N/2+1:N)$ as the corresponding output;
```

Algorithm 3: An $O(N\log^2 N)$ recursive algorithm for solving a lower triangular system $Lx = b$ when L is stored in a format of H-IDBF. The notation "$-=$" in Line 5 stands for the updating operator defined as: $a -= b$ is equivalent to $a = a - b$ for any array a and b.

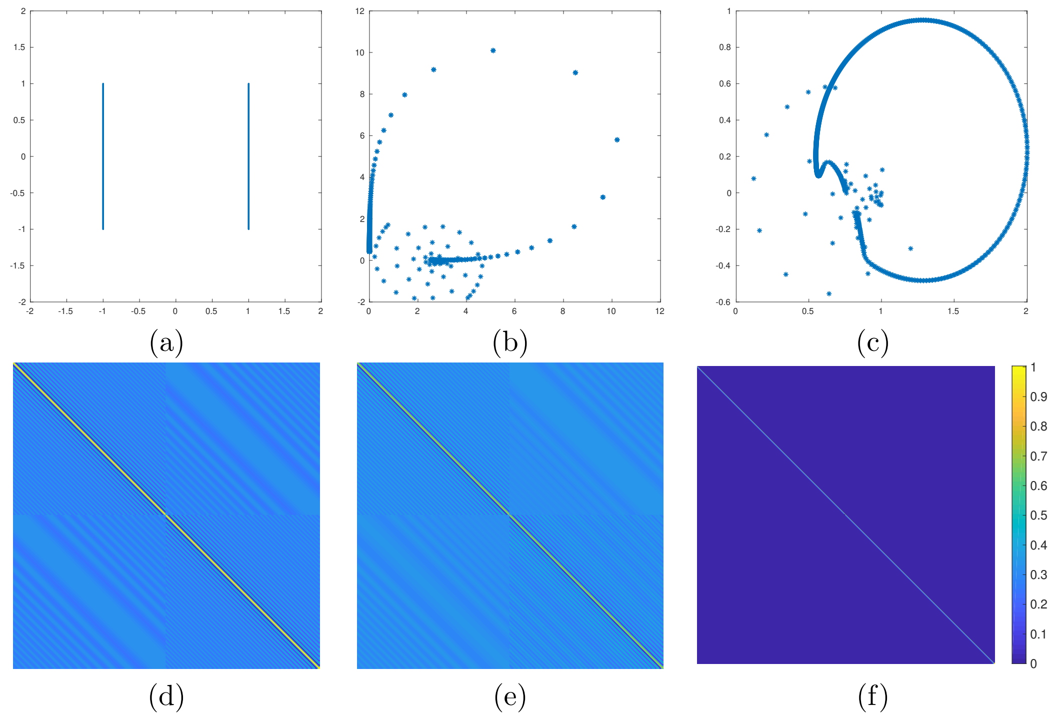

Figure 7: (a) Two parallel strips. (b) The eigenvalue distribution of the impedance matrix A corresponding to (a). (c) The eigenvalue distribution of $\tilde{L}^{-1}A\tilde{U}^{-1}$, the impedance matrix after preconditioning. (d) The real part of A. (e) The real part of $L+U-I$, where L and U are the LU factors of A, and I is an identity matrix. (f) The magnitude of the entries of $\tilde{L}^{-1}A\tilde{U}^{-1}$.

```
  Data: A H-IDBF of $U \in \mathbb{C}^{N\times N}$ and a vector $b \in \mathbb{C}^N$.
  Result: A vector $x \in \mathbb{C}^N$ equal to $U^{-1}b$.
1 if $N \leq n_0$ then
2     Solve the triangualr system $Ux = b$ via standard LAPACK;
3 else
4     Partition the upper triangular system as
          $$U = \begin{pmatrix} F_{11} & F_{12} \\ 0 & F_{22} \end{pmatrix};$$
      Recursively apply Algorithm 4 with $F_{22}$ and $b(N/2+1:N)$ as the input to obtain
      $x(N/2+1:N)$ as the corresponding output;
5     Apply Algorithm 2 with $-F_{12}$ to obtain $b(1:N/2) -= F_{12}x(1+N/2:N)$;
6     Recursively apply Algorithm 4 with $F_{11}$ and $b(1:N/2)$ as the input to obtain $x(1:N/2)$
      as the corresponding output;
```

Algorithm 4: An $O(N\log^2 N)$ recursive algorithm for solving an upper triangular system $Ux = b$ when U is stored in a format of H-IDBF. The notation "$-=$" in Line 5 stands for the updating operator defined as: $a -= b$ is equivalent to $a = a - b$ for any array a and b.

4 Numerical results

This section demonstrates the efficiency and accuracy of the proposed preconditioner via its application to wide classes of open surfaces: a semicircle, a corrugated corner reflector, a spiral line, two parallel strips, an open square, and a cup-shaped cavity. In all examples, the surfaces are discretized with approximately 20 pulse basis functions per wavelength. The leaf sizes in the dyadic trees of each butterfly-compressed block are set to approximately $n_0 = 200$. The compression tolerance in linear scaling ID is set to ϵ=1.0E-4, and the oversampling parameter t in the linear scaling ID is set to 1. For the aforementioned surfaces, the maximum butterfly ranks k among all blocks of the impedance matrices are respectively 7, 11, 10, 7, 14, 13, which are independent of the matrix sizes N. The matrix entries are scaled with a scalar such that the largest diagonal entry has unit magnitude. A transpose-free quasi-minimal residual (TFQMR) iterative solver is utilized with a convergence tolerance 1.0E-5. All experiments are performed on the Cori Haswell machine at NERSC, which is a Cray XC40 system and consists of 2388 dual-socket nodes with Intel Xeon E5-2698v3 processors running 16 cores per socket. The nodes are configured with 128 GB of DDR4 memory at 2133 MHz.

First, the accuracy of the proposed preconditioner is demonstrated by changing the matrix size N from 5000 to 5,000,000. Let x^t be a randomly generated $N \times 1$ vector, the right hand side is generated as $b = Ax^t$. Note that A is replaced by its H-IDBF compression F when $N > 10000$. Let x^a denote the solution vector computed from the preconditioned linear system. The solution error is defined as $\epsilon^a = \left\|x^a - x^t\right\|_2 / \left\|x^t\right\|_2$. From Table 1, reasonably good accuracy has been observed for all classes of surfaces. However, a slight degradation in accuracy when N increases is also observed.

Next, the computational efficiency is demonstrated by investigating the computation time, memory, and iteration counts as matrix size N increases. The construction time, iterative solution time, and overall memory are plotted in Figure 8 for all test surfaces. It can be easily validated that all three quantities scale as $O(N \log^2 N)$ as predicted. It is worth mentioning that for the semicircle example, the proposed preconditioner permits rapid solution for $N = 100$ million, which is very competitive to the state-of-the-art MLFMA-based iterative solvers. We also observe that the solution time (i.e., application and triangular solve of H-IDBF-LU in the iterative solver) dominates the overall computation time especially for surfaces that requires constant but high number of iterations. We then compare the iteration counts of the proposed preconditioner and an iterative solver without any preconditioner (see Figure 9). The iteration counts required by the iterative solver without preconditioner grow rapidly for all surfaces (dashed lines in Figure 9). In contrast, For surfaces including the semicircle, spiral lines and corrugated corners, the iteration counts using the proposed preconditioner stay as small constant (typically less than 30); for the other surfaces, the observed iteration counts scale at most as $O(\log^2 N)$. It's worth mentioning that for complicated geometries such as highly resonant cavities or closed surfaces, the iteration counts can grow much faster.

shape	semicircle	corner	spiral	strips	square	cup
N=5E3	2.24E-06	9.51E-06	8.13E-06	7.12E-05	2.28E-05	1.60E-05
N=5E4	1.11E-05	9.84E-06	3.82E-05	6.45E-04	2.24E-04	1.84E-04
N=5E5	5.86E-06	3.85E-06	4.01E-05	9.86E-04	2.38E-04	3.56E-04
N=5E6	1.10E-05	8.17E-06	1.37E-04	3.74E-04	2.33E-04	6.16E-04

Table 1: Measured solution error for different geometry shapes.

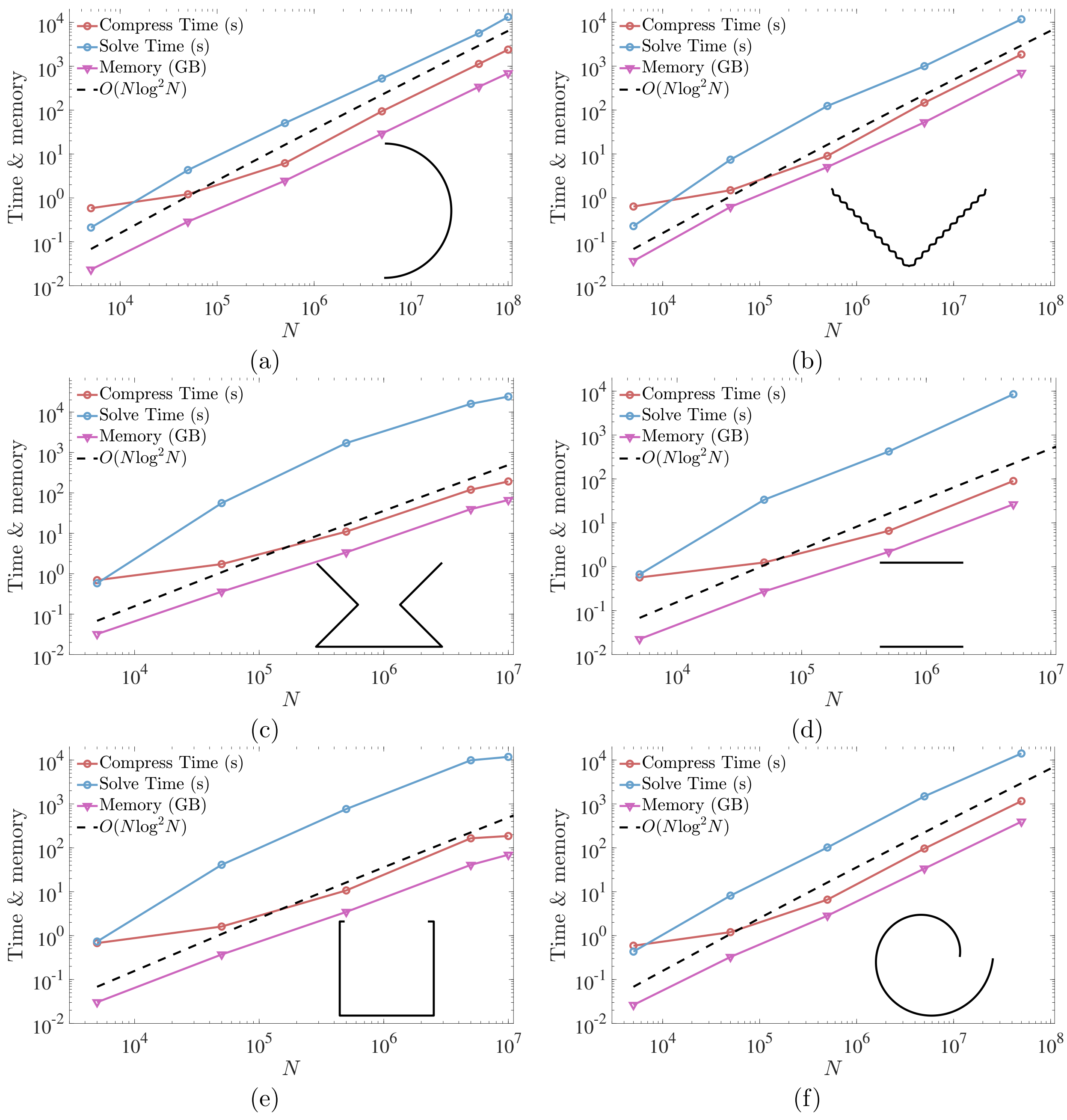

Figure 8: Computation time and memory of the proposed preconditioner for (a) a semicircle, (b) a corrugated corner, (c) a cup-shaped cavity, (d) two parallel strips, (e) an open square, and (f) a spiral line.

5 Conclusion and discussion

This paper has introduced a simple and efficient algorithm, the hierarchical interpolative decomposition butterfly-LU factorization (H-IDBF-LU) preconditioner for solving two-dimensional electric-field integral equations (EFIEs) in electromagnetic scattering problems of perfect electrically conducting objects with open surfaces. H-IDBF-LU consists of two main parts: the first part applies

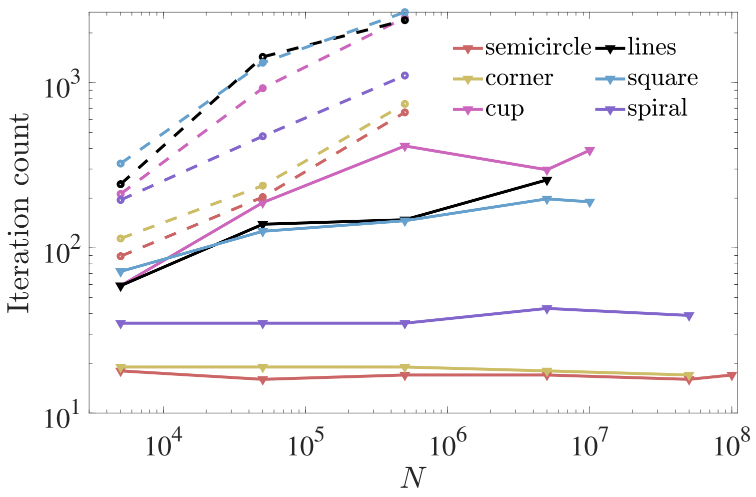

Figure 9: Iteration counts with and without the proposed preconditioners for different arc shapes.

the newly developed interpolative decomposition butterfly factorization (IDBF) to compress dense blocks of the discretized EFIE operator to expedite its application; the second part treats the lower and upper triangular part of the IDBF as an approximate LU factorization of the EFIE operator leading to an efficient preconditioner in iterative solvers.

Both the memory requirement and computational cost of the H-IDBF-LU solver scale as $O(N \log^2 N)$ in one iteration; the total number of iterations required for a reasonably good accuracy scales as $O(1)$ to $O(\log^2 N)$ in all of our numerical tests. Our algorithm is simple to implement, automatically adapts to different structures with open surfaces, and is competitive with state-of-the-art MLFMA-based algorithms. A user-friendly MATLAB package, ButterflyLab (`https://github.com/ButterflyLab/ButterflyLab`), and a distributed parallel Fortran/C++ package, ButterflyPack (`https://github.com/liuyangzhuan/ButterflyPACK`), are freely available online.

The lower and upper triangular part of the EFIE oerator can serve as an approximate LU factorization of the EFIE operator is an interesting observation and deserves much theoretical attention in the future. This idea could also be applied to three-dimensional EFIE's with open surfaces and we will explore this in a future work.

Acknowledgments. H. Yang thanks the support of the start-up grant by the Department of Mathematics at the National University of Singapore, and the support of the Ministry of Education in Singapore under the grant MOE2018-T2-2-147. Y. Liu thanks the support from the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program through the FASTMath Institute under Contract No. DE-AC02-05CH11231 at Lawrence Berkeley National Laboratory. The authors also thank Dr. Xiaoye Sherry Li for useful discussions and suggestions on the paper.

References

[1] Vladimir Rokhlin. Rapid solution of integral equations of scattering theory in two dimensions. *Journal of Computational Physics*, 86(2):414 – 439, 1990.

[2] Vladimir Rokhlin. Diagonal forms of translation operators for the helmholtz equation in three dimensions. *Applied and Computational Harmonic Analysis*, 1(1):82 – 93, 1993.

[3] Hongwei Cheng, William Y. Crutchfield, Zydrunas Gimbutas, Leslie F. Greengard, J. Frank Ethridge, Jingfang Huang, Vladimir Rokhlin, Norman Yarvin, and Junsheng Zhao. A wideband fast multipole method for the helmholtz equation in three dimensions. *Journal of Computational Physics*, 216(1):300 – 325, 2006.

[4] Ronald Coifman, Vladimir Rokhlin, and Stephen Wandzura. The fast multipole method for the wave equation: a pedestrian prescription. *IEEE Antennas and Propagation Magazine*, 35(3):7–12, June 1993.

[5] Eric Darve. The fast multipole method i: Error analysis and asymptotic complexity. *SIAM Journal on Numerical Analysis*, 38(1):98–128, 2000.

[6] Michael A. Epton and Benjamin Dembart. Multipole translation theory for the three-dimensional laplace and helmholtz equations. *SIAM Journal on Scientific Computing*, 16(4):865–897, 1995.

[7] Jiming Song, Cai-Cheng Lu, and Weng Cho Chew. Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects. *IEEE Transactions on Antennas and Propagation*, 45(10):1488–1493, Oct 1997.

[8] Lexing Ying. Fast directional computation of high frequency boundary integrals via local ffts. *Multiscale Modeling & Simulation*, 13(1):423–439, 2015.

[9] Björn Engquist and Lexing Ying. A fast directional algorithm for high frequency acoustic scattering in two dimensions. *Commun. Math. Sci.*, 7(2):327–345, 06 2009.

[10] Björn Engquist and Lexing Ying. Fast directional multilevel algorithms for oscillatory kernels. *SIAM Journal on Scientific Computing*, 29(4):1710–1737, 2007.

[11] Matthias Messner, Martin Schanz, and Eric Darve. Fast directional multilevel summation for oscillatory kernels based on chebyshev interpolation. *Journal of Computational Physics*, 231(4):1175 – 1196, 2012.

[12] Amir Z. Averbuch, Elena Braverman, Ronald R. Coifman, Moshe Israeli, and Avram Sidi. Efficient computation of oscillatory integrals via adaptive multiscale local fourier bases. *Applied and Computational Harmonic Analysis*, 9(1):19 – 53, 2000.

[13] Brian D. Bradie, Ronald R. Coifman, and Alexander Grossmann. Fast numerical computations of oscillatory integrals related to acoustic scattering, i. *Applied and Computational Harmonic Analysis*, 1(1):94 – 99, 1993.

[14] Hai Deng and Hao Ling. Fast solution of electromagnetic integral equations using adaptive wavelet packet transform. *IEEE Transactions on Antennas and Propagation*, 47(4):674–682, April 1999.

[15] Francis X. Canning. Sparse approximation for solving integral equations with oscillatory kernels. *SIAM Journal on Scientific and Statistical Computing*, 13(1):71–87, 1992.

[16] Laurent Demanet and Lexing Ying. Wave atoms and sparsity of oscillatory patterns. *Applied and Computational Harmonic Analysis*, 23(3):368 – 387, 2007.

[17] Eric Michielssen and Amir Boag. A multilevel matrix decomposition algorithm for analyzing scattering from large structures. *Antennas and Propagation, IEEE Transactions on*, 44(8):1086–1093, Aug 1996.

[18] Michael O'Neil, Franco Woolfe, and Vladimir Rokhlin. An algorithm for the rapid evaluation of special function transforms. *Appl. Comput. Harmon. Anal.*, 28(2):203–226, 2010.

[19] Yingzhou Li, Haizhao Yang, Eileen R. Martin, Kenneth L. Ho, and Lexing Ying. Butterfly Factorization. *Multiscale Modeling & Simulation*, 13(2):714–732, 2015.

[20] Qiyuan Pang, Kenneth L. Ho, and Haizhao Yang. Interpolative decomposition butterfly factorization. *arXiv:1809.10573 [math.NA]*, 2018.

[21] Yingzhou Li, Haizhao Yang, and Lexing Ying. Multidimensional butterfly factorization. *Applied and Computational Harmonic Analysis*, 2017.

[22] Haizhao Yang. A unified framework for oscillatory integral transform: When to use NUFFT or butterfly factorization? *arXiv:1803.04128 [math.NA]*, 2018.

[23] Xavier Antoine, Abderrahmane Bendali, and Marion Darbas. Analytic preconditioners for the electric field integral equation. *International Journal for Numerical Methods in Engineering*, 61(8):1310–1331.

[24] Snorre H. Christiansen and Jean-Claude Nédélec. A preconditioner for the electric field integral equation based on Caldéron formulas. *SIAM Journal on Numerical Analysis*, 40(3):1100–1135, 2002.

[25] Oscar P. Bruno and Stéphane K. Lintner. Second-kind integral solvers for TE and TM problems of diffraction by open arcs. *Radio Science*, 47, 2012.

[26] Lexing Ying. Directional preconditioner for 2D high frequency obstacle scattering. *Multiscale Modeling & Simulation*, 13(3):829–846, 2015.

[27] Jian-Gong Wei, Zhen Peng, and Jin-Fa Lee. A fast direct matrix solver for surface integral equation methods for electromagnetic wave scattering from non-penetrable targets. *Radio Science*, 47(5).

[28] Leslie Greengard, Denis Gueyffier, Per-Gunnar Martinsson, and Vladimir Rokhlin. Fast direct solvers for integral equations in complex three-dimensional domains. *Acta Numer.*, 18:243–275, 2009.

[29] Alex Heldring, Juan M. Rius, José M. Tamayo, Josep Parrn, and Eduard Ubeda. Multiscale compressed block decomposition for fast direct solution of method of moments linear system. *IEEE Transactions on Antennas and Propagation*, 59(2):526–536, Feb 2011.

[30] John Shaeffer. Direct solve of electrically large integral equations for problem sizes to 1 M unknowns. *IEEE Transactions on Antennas and Propagation*, 56(8):2306–2313, Aug 2008.

[31] Wenwen Chai and Dan Jiao. An $\mathcal{H}^2$-matrix-based integral-equation solver of reduced complexity and controlled accuracy for solving electrodynamic problems. *IEEE Transactions on Antennas and Propagation*, 57(10):3147–3159, Oct 2009.

[32] Mario Bebendorf. Hierarchical LU decomposition-based preconditioners for bem. *Computing*, 74(3):225–247, May 2005.

[33] Victor Minden, Kenneth Ho, Anil Damle, and Lexing Ying. A recursive skeletonization factorization based on strong admissibility. *Multiscale Modeling & Simulation*, 15(2):768–796, 2017.

[34] Eduardo Corona, Per-Gunnar Martinsson, and Denis Zorin. An o(n) direct solver for integral equations on the plane. *Applied and Computational Harmonic Analysis*, 38(2):284 – 317, 2015.

[35] Per Gunnar Martinsson and Vladimir Rokhlin. A fast direct solver for scattering problems involving elongated structures. *Journal of Computational Physics*, 221(1):288 – 302, 2007.

[36] Eric Michielssen, Amir Boag, and Weng Cho Chew. Scattering from elongated objects: direct solution in O(N log/sup 2/ N) operations. *IEE Proceedings - Microwaves, Antennas and Propagation*, 143(4):277–283, Aug 1996.

[37] Emil Winebrand and Amir Boag. A multilevel fast direct solver for em scattering from quasi-planar objects. In *2009 International Conference on Electromagnetics in Advanced Applications*, pages 640–643, Sept 2009.

[38] Yaniv Brick, Vitaliy Lomakin, and Amir Boag. Fast direct solver for essentially convex scatterers using multilevel non-uniform grids. *IEEE Transactions on Antennas and Propagation*, 62(8):4314–4324, Aug 2014.

[39] Yang Liu, Han Guo, and Eric Michielssen. An HSS matrix-inspired butterfly-based direct solver for analyzing scattering from two-dimensional objects. *IEEE Antennas and Wireless Propagation Letters*, 16:1179–1183, 2017.

[40] Han Guo, Yang Liu, Jun Hu, and Eric Michielssen. A butterfly-based direct integral-equation solver using hierarchical LU factorization for analyzing scattering from electrically large conducting objects. *IEEE Transactions on Antennas and Propagation*, 65(9):4742–4750, Sept 2017.

[41] Han Guo, Yang Liu, Jun Hu, and Eric Michielssen. A butterfly-based direct solver using hierarchical LU factorization for Poggio-Miller-Chang-Harrington-Wu-Tsai equations. *Microw Opt Technol Lett.*, 60:13811387, 2018.

[42] Peter Hoffman and K. C. Reddy. Numerical differentiation by high order interpolation. *SIAM Journal on Scientific and Statistical Computing*, 8(6):979–987, 1987.

[43] John P. Boyd and Fei Xu. Divergence (Runge Phenomenon) for least-squares polynomial approximation on an equispaced grid and Mock Chebyshev subset interpolation. *Applied Mathematics and Computation*, 210(1):158 – 168, 2009.

[44] George Neville Watson. *A Treatise on the Theory of Bessel Functions. Second edition.* 1995.

[45] Emmanuel J. Candès, Laurent Demanet, and Lexing Ying. A fast butterfly algorithm for the computation of Fourier integral operators. *Multiscale Modeling and Simulation*, 7(4):1727–1750, 2009.

[46] Steffen Börm, Lars Grasedyck, and Wolfgang Hackbusch. *Hierarchical matrices.* 2005.

[47] Lars Grasedyck and Wolfgang Hackbusch. Construction and arithmetics of H-matrices. *Computing*, 70(4):295–334, Aug 2003.

[48] Lin Lin, Jianfeng Lu, and Lexing Ying. Fast construction of hierarchical matrix representation from matrix-vector multiplication. *J. Comput. Phys.*, 230(10):4071–4087, 2011.

[49] Per-Gunnar Martinsson. A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix. *SIAM J. Matrix Anal. Appl.*, 32(4):1251–1274, 2011.