Interpolative Decomposition Butterfly Factorization

Qiyuan Pang
Tsinghua University, China
ppangqqyz@foxmail.com

Kenneth L. Ho
San Francisco, CA, USA
klho@alumni.caltech.edu

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

September 27, 2018

Abstract

This paper introduces an interpolative decomposition butterfly factorization (IDBF) as a data-sparse approximation for matrices that satisfy a complementary low-rank property. The IDBF can be constructed in $O(N \log N)$ operations for an $N \times N$ matrix via hierarchical interpolative decompositions (IDs), if matrix entries can be sampled individually and each sample takes $O(1)$ operations. The resulting factorization is a product of $O(\log N)$ sparse matrices, each with $O(N)$ non-zero entries. Hence, it can be applied to a vector rapidly in $O(N \log N)$ operations. Numerical results are provided to demonstrate the effectiveness of the butterfly factorization and its construction algorithms.

Keywords. Data-sparse matrix, butterfly factorization, interpolative decomposition, operator compression, Fourier integral operators, special functions, high-frequency integral equations.

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

1 Introduction

One of the key computational task in scientific computing is to evaluate dense matrix-vector multiplication (matvec) rapidly. Given a dense matrix $K \in \mathbb{C}^{N\times N}$ and a vector $x \in \mathbb{C}^N$, it takes $O(N^2)$ operations to naively compute the vector $y = Kx \in \mathbb{C}^N$. There has been extensive research in constructing data-sparse representation of structured matrices (e.g., low-rank matrices [1, 2, 3, 4, 5], $\mathcal{H}$ matrices [6, 7, 8], $\mathcal{H}^2$ matrics [9, 10], HSS matrices [11, 12], complementary low-rank matrices [13, 14, 15, 16, 17, 18], FMM [19, 20, 21, 22, 23, 24, 25, 26], directional low-rank matrices [27, 28, 29, 30], and the combination of these matrices [31, 32]) aiming for linear or nearly linear scaling matvec. In particular, this paper concerns nearly optimal matvec for complementary low-rank matrices.

A wide range of transforms in harmonic analysis [14, 15, 33, 34, 35, 36], and integral equations in the high-frequency regime [31, 32] admit a matrix or its submatrices satisfying a complementary low-rank property. For a complementary low-rank matrix, its rows are typically indexed by a point set X, and the columns by another point set Ω, both X and Ω are point sets in $\mathbb{R}^1$. Associated with X and Ω are two trees T_X and T_Ω constructing by dyadic partition. Both trees have the same level $L = O(\log N)$, with the top root being the 1-th level and the bottom leaf level being the L-th level. We say a matrix satisfies the complementary low-rank property if, any node A at level l in

T_X, and any node B at level $L-l$, the submatrix $K^l_{A,B}$ of K, obtained by restricting the rows of K to the points in node A and the columns to the points in node B, was numerically low-rank; that is, given a precision ϵ, there exists an approximation of $K^l_{A,B}$ with the 2-norm of the error bounded by ϵ and the rank k bounded by a polynomial in $\log N$ and $\log 1/\epsilon$. See Figure 1 for an illustration of low-rank submatrices in a complementary low-rank matrix of size 16×16.

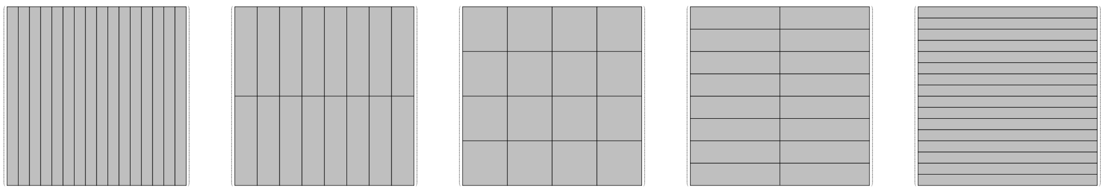


Figure 1: Hierarchical decomposition of the row and column indices of a 16×16 matrix. The dyadic trees T_X and T_Ω have roots containing 16 rows and 16 columns respectively, and their leaves containing only a single row or column. The partition above indicates the complementary low-rank property of the matrix, and assumes that each submatrix is rank-1.

This paper introduces an **Interpolative Decomposition Butterfly Factorization (IDBF)** as a data-sparse approximation for matrices that satisfy the complementary low-rank property. The IDBF can be constructed in $O(\frac{k^3}{n_0} N \log N)$ operations for an $N \times N$ matrix K with a local rank parameter k and a leaf size parameter n_0 via hierarchical linear interpolative decompositions (IDs), if matrix entries can be sampled individually and each sample takes $O(1)$ operations. The resulting factorization is a product of $O(\log N)$ sparse matrices, each of which contains $O(\frac{k^2}{n_0} N)$ nonzero entries as follows:

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

where $h = L/2$ and the level L is assumed to be even. Hence, it can be applied to a vector rapidly in $O(\frac{k^2}{n_0} N \log N)$ operations. Previously, purely algebraic butterfly factorizations (in the sense that the complementary matrix is not the discretization of a kernel function $K(x,\xi) = a(x,\xi)e^{2\pi i \Phi(x,\xi)}$ with smooth $a(x,\xi)$ and $\Phi(x,\xi)$) have at least $O(N^{1.5})$ scaling [13, 14, 15, 17]. The IDBF is the first **purely algebraic butterfly factorization** (BF) with $O(N \log N)$ scaling in both factorization and application.

2 Interpolative Decomposition Butterfly Factorization (IDBF)

We will describe IDBF in detail in this section. For the sake of simplicity, we assume that $N = 2^L n_0$, where L is an even integer, and $n_0 = O(1)$ 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. Let's briefly introduce the main ideas of designing $O(\frac{k^3}{n_0} N \log N)$ IDBF using a linear ID. In IDBF, we compute $O(\log N)$ levels of low-rank submatrix factorizations. At each level, according to the matrix partition by the dyadic trees in column and row (see Figure 1 for an example), there are $\frac{N}{n_0}$ low-rank submatrices. At the ℓ-th level, the size of submatrices to be factorized is $\frac{N}{2^\ell} \times n$ or $n \times \frac{N}{2^\ell}$, where $n \leq 2k$ and k is the rank parameter of IDBF. Linear IDs require $O(k^2 n) \lesssim O(k^3)$ operation for each submatrix, and hence at most $O(\frac{k^3}{n_0} N)$ for each level of factorization, and $O(\frac{k^3}{n_0} N \log N)$ for the whole IDBF. There are two differences between IDBF and other BFs [13, 14, 15].

1. The order of factorizations is from the leaf-root and root-leaf levels of matrix partitioning

(e.g., the left and right panels in Figure 1) and moves towards the middle level of matrix partitioning (e.g., the middle panel of Figure 1).

2. Linear IDs are organized in an appropriate way such that it is cheep in terms of both memory and operations to provide all necessary information for each level of factorizations.

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 $|c|$. Given a matrix A, A_{pq} or $A_{p,q}$ is the submatrix with rows and columns restricted to the index sets p and q, respectively. We also use the MATLAB® notation $A_{:,q}$ to denote the submatrix with columns restricted to q.

2.1 Linear scaling Interpolative Decompositions

Interpolative decomposition and other low-rank decomposition techniques [1, 3, 37] are important elements in modern scientific computing. These techniques usually require $O(mn)$ arithmetic operations to get a rank $k = O(1)$ matrix factorization to approximate a matrix $A \in \mathbb{C}^{m\times n}$. Linear scaling randomized techniques can reduce the cost to $O(m + n)$ [38]. [39] further shows that in the CUR low-rank approximation $A \approx CUR$, where $C = A_{:,c}$, $R = A_{r,:}$, and $U \in \mathbb{C}^{k\times k}$ with $|c| = |r| = k$, if only U, c, and r are needed, there exists an $O(k^3)$ algorithm for constructing U, c, and r.

In the construction of IDBF, we use an $O(nk^2)$ linear scaling column ID to construct T and select skeleton indices c such that $A \approx A_{:,c}T$ when $n \ll m$. Similarly, we can construct a row ID $A \approx TA_{r,:}$ in $O(mk^2)$ operations when $m \ll n$. Either randomized sampling or Mock-Chebyshev grids[1] [41, 42] can be applied to reduce the quadratic computational cost to linear. Here we present a simple lemma of interpolative decomposition (ID) to motivate the proposed linear scaling ID.

Lemma 2.1. *For a matrix $A \in \mathbb{C}^{m\times n}$ with rank $k \le \min\{m, n\}$, there exists a partition of the column indices of A, $p \cup q$ with $|q| = k$, and a matrix $T \in \mathbb{C}^{k\times n}$, such that $A_{:,p} = A_{:,q}T$.*

Proof. A rank revealing QR decomposition of A gives

$$A\Lambda = QR = Q[R_1\ R_2], \tag{2}$$

where $Q \in \mathbb{C}^{m\times k}$ is an orthogonal matrix, $R \in \mathbb{C}^{k\times n}$ is upper triangular, 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. Let

$$A_{:,q} = QR_1, \tag{3}$$

and then

$$A_{:,p} = QR_2 = QR_1R_1^{-1}R_2 = A_{:,q}T, \tag{4}$$

where

$$T = R_1^{-1}R_2. \tag{5}$$

□

[1] Though it was shown in [40] that no fast stable approximation of analytic functions from equispaced samples in a bounded interval in the sense of L^∞-norm with an exponential convergence rate is available, the Mock-Chebyshev points admit polynomial interpolation with a root-exponential convergence rate. In this paper, we care more about the approximation error at the equispaced sampling locations, in which case it is still unknown whether the Mock-Chebyshev points admit an exponential convergence rate.

$A_{:,p} = A_{:,q}T$ in Lemma 2.1 is equivalent to the traditional form of a column ID,

$$A = A_{:,q}[I\ T]\Lambda := A_{:,q}V, \tag{6}$$

where Λ is a permutation matrix associated with p and q. We call p and q as *skeleton* and *redundant* indices, respectively. V can be understood as a *column interpolation matrix*. Our goal for linear scaling ID is to construct the skeleton index set q, the redundant index set p, T, and Λ in $O(k^2n)$ operations and $O(kn)$ memory.

For a tall skinny matrix A, i.e., $m \gg n$, the rank revealing QR decomposition of A in (2) typically requires $O(mn^2)$ operations. To reduce the complexity to $O(k^2n)$, we actually apply the rank revealing QR decomposition to pk carefully selected rows of A, where p is an oversampling number. These columns can be chosen independently and uniformly from the row space as the sublinear CUR in [39] or the linear scaling algorithm in [38]; or they can be chosen from the Mock-Chebyshev grids of the row indices as in [41, 42, 18]. After the rank revealing QR decomposition, the other steps to generate T and Λ take only $O(k^2n)$ operations since R_1 in (5) is an upper triangular matrix. We refer this linear scaling column ID as *cID*.

For a short and fat matrix $A \in \mathbb{C}^{m\times n}$ with $m \ll n$, a similar row ID

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

can be devised similarly with $O(k^2m)$ operations and $O(km)$ memory, where * denotes the conjugate transpose of a matrix. We refer this linear scaling row ID as *rID* and U as the *row interpolation matrix*.

2.2 Leaf-root complementary skeletonization (LRCS)

For a complementary low-rank matrix A, we introduce the *leaf-root complementary skeletonization (LRCS)*

$$A \approx USV$$

via *cID*s of the submatrices corresponding to the leaf-root levels of the column-row dyadic trees (e.g., see the associated matrix partition in Figure 2 (right)), and *rID*s of the submatrices corresponding to the root-leaf levels of the column-row dyadic trees (e.g., see the associated matrix partition in Figure 2 (middle)). We always assume that IDs in this Section is applied with a rank parameter $k = O(1)$. We'll not specify k again in the following discussion.

Suppose 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 partitioned into leaves $\{r_i\}_{1\le i\le m}$ (and $\{c_i\}_{1\le i\le m}$) as follows

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

with $|r_i| = n_0$ (and $|c_i| = n_0$) for all $1 \le i \le m$, where $m = 2^L = \frac{N}{n_0}$, and $L = \log_2 N - \log_2 n_0$ is the depth of the dyadic trees T_X (and T_Ω). Figure 2 shows an example of row and column dyadic trees with $m = 16$. We apply *rID* to each $A_{r_i,:}$ to obtain the row interpolation matrix in its ID and denote it as U_i; the associated skeleton indices of the ID is denoted as $\hat{r}_i \subset r_i$. Let

$$\hat{r} = [\hat{r}_1, \hat{r}_2, \cdots, \hat{r}_m], \tag{9}$$

then $A_{\hat{r},:}$ is the important skeleton of A and we have

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

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

$$A \approx \begin{pmatrix} U_1 & & & \\ & U_2 & & \\ & & \ddots & \\ & & & U_m \end{pmatrix} \begin{pmatrix} A_{\hat{r}_1,\hat{c}_1} & A_{\hat{r}_1,\hat{c}_2} & \dots & A_{\hat{r}_1,\hat{c}_m} \\ A_{\hat{r}_2,\hat{c}_1} & A_{\hat{r}_2,\hat{c}_2} & \dots & A_{\hat{r}_2,\hat{c}_m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{\hat{r}_m,\hat{c}_1} & A_{\hat{r}_m,\hat{c}_2} & \dots & A_{\hat{r}_m,\hat{c}_m} \end{pmatrix} \begin{pmatrix} V_1 & & & \\ & V_2 & & \\ & & \ddots & \\ & & & V_m \end{pmatrix} := USV. \quad (10)$$

For a concrete example, Figure 3 visualizes the non-zero pattern of the LRCS in (10) of the complementary low-rank matrix A in Figure 2.

The novelty of the LRCS is that M and S are not computed explicitly; instead, they are generated and stored via the skeleton of row and column index sets. 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 (10), since there are $2m = \frac{2N}{n_0}$ IDs in total.

It is worth emphasizing that in the LRCS of a complementary matrix $A \approx USV$, the matrix S is again a complementary matrix. The row (and column) dyadic tree $\hat{T}_X$ (and $\hat{T}_\Omega$) of S is the compressed version of the row (and column) dyadic trees T_X (and T_Ω) of A. Figure 4 (or 5) visualizes the relation of T_X and $\hat{T}_X$ (or T_Ω and $\hat{T}_\Omega$) for the complementary matrix A in Figure 2. $\hat{T}_X$ (or $\hat{T}_\Omega$) is not compressible at the leaf level of T_X (or T_Ω) but it is compressible if it is considered as a dyadic tree with one depth less (see Figure 6 for an example of a new compressible dyadic tree with one depth less).

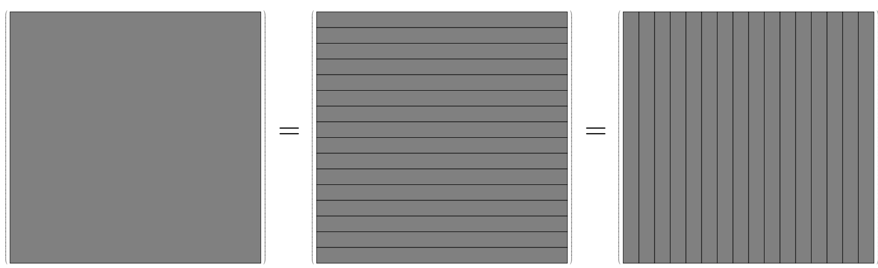


Figure 2: The left matrix is a complementary low-rank matrix. Assume that the depth of the dyadic trees of column and row spaces is 5. The middle figure visualizes the root-leaf partitioning that divides the row index set into sixteen continuous subsets as sixteen leaves. The right one is for the leaf-root partitioning that divides the column index set into sixteen continuous subsets as sixteen leaves.

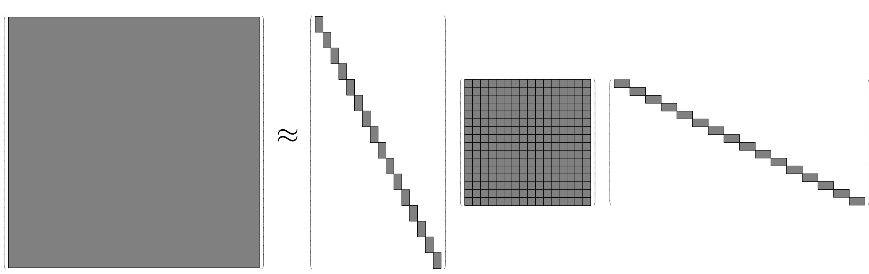


Figure 3: An example of the LRCS in (10) of the complementary low-rank matrix A in Figure 2. Non-zero submatrices in (10) are shown in gray areas.

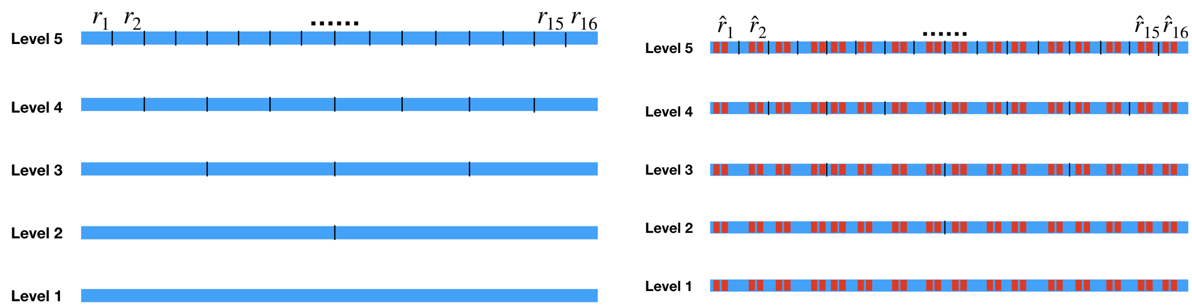


Figure 4: Left: The dyadic tree T_X of the row space with leaves $\{r_i\}_{1\leq i\leq 16}$ denoted as in (8) for the example in Figure 2. Right: Selected important rows of T_X naturally form a compressed dyadic tree in red with leaves $\{\hat{r}_i\}_{1\leq i\leq 16}$ denoted as in (9).

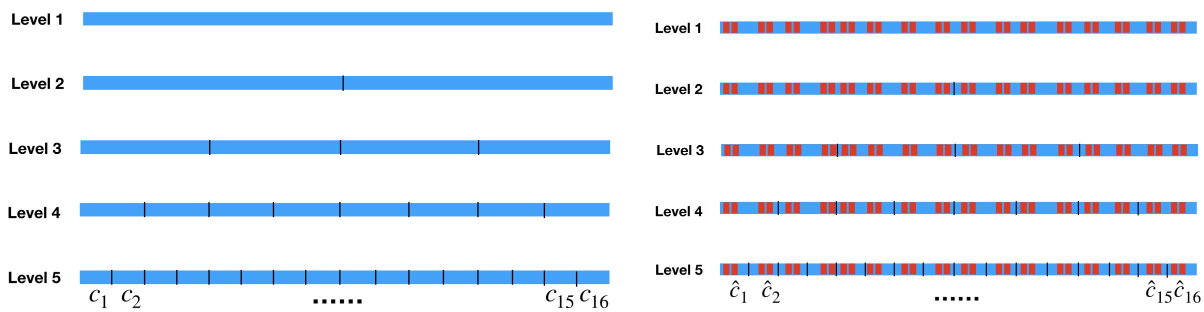


Figure 5: Left: The dyadic tree T_Ω of the column space with leaves $\{c_i\}_{1\leq i\leq 16}$ denoted as in (8) for the example in Figure 2. Right: Selected important columns of T_Ω naturally form a compressed dyadic tree in red with leaves $\{\hat{c}_i\}_{1\leq i\leq 16}$.

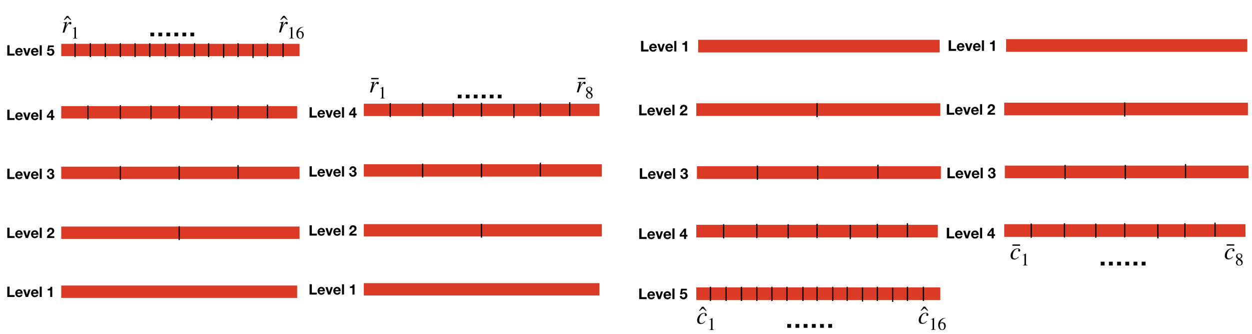


Figure 6: Left: The compressed dyadic tree of T_X of the row space in Figure 4. Level 5 is not compressible. Middle left: Combining adjacent leaves at Level 5, i.e., $\bar{r}_i = \hat{r}_{2i-1} \cup \hat{r}_{2i}$, forms a compressible dyadic tree with depth 4. Middle right: the compressed dyadic tree of T_Ω of the column space in Figure 5. Level 5 is not compressible. Right: Combining adjacent leaves at Level 5, i.e., $\bar{c}_i = \hat{c}_{2i-1} \cup \hat{c}_{2i}$, forms a compressible dyadic tree with depth 4.

2.3 Matrix splitting with complementary skeletonization (MSCS)

Here we describe another elementary idea of IDBF that is applied repeatedly: MSCS. A complementary low-rank matrix A (with row and column dyadic trees T_X and T_Ω of depth L and with $m = 2^L$ leaves) can be split into a 2×2 block matrix

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \tag{11}$$

according to the nodes of the second level of the dyadic trees T_X and T_Ω (those nodes right next to the root level). By the complementary low-rank property of A, we know that A_{ij} is also complementary low-rank, for all i and j, with row and column dyadic trees $T_{X,ij}$ and $T_{\Omega,ij}$ of depth $L-1$ and with $m/2$ leaves.

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

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

The factorization in (12) is referred as the *matrix splitting with complementary skeletonization (MSCS)* in this paper. Recall that the middle factor S is not explicitly computed, resulting in a linear scaling algorithm for forming (12). Figure 7 visualizes the MSCS of a complementary low-rank matrix A with dyadic trees of depth 5 and 16 leaf nodes in Figure 2.

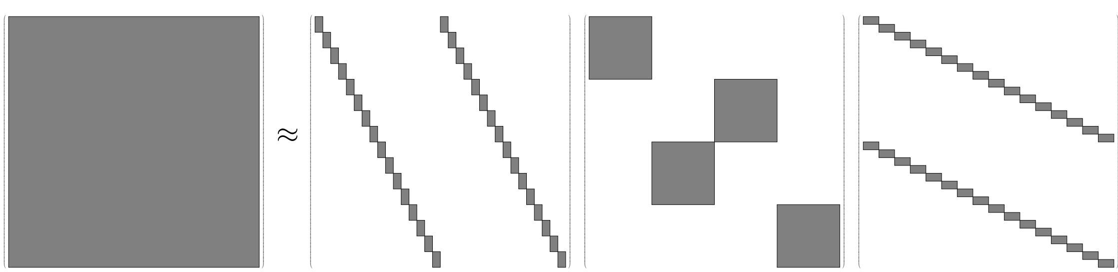


Figure 7: The visualization of a MSCS of a complementary low-rank matrix $A \approx USV$ with dyadic trees of depth 5 and 16 leaf nodes in Figure 2. Non-zero blocks in (12) are shown in gray areas.

2.4 Recursive MSCS

Now we apply MSCS recursively to get the full IDBF of a complementary low-rank matrix A (with row and column dyadic trees T_X and T_Ω of depth L and with $m = 2^L$ leaves). As in (12), suppose we have constructed the first level of MSCS and denote it as

$$A = U^L S^L V^L \tag{13}$$

with

$$
\begin{aligned}
U^L &= \begin{pmatrix} U_{11}^L & & U_{12}^L & \\ & U_{21}^L & & U_{22}^L \end{pmatrix}, \\
S^L &= \begin{pmatrix} S_{11}^L & & & \\ & & S_{21}^L & \\ & S_{12}^L & & \\ & & & S_{22}^L \end{pmatrix}, \\
V^L &= \begin{pmatrix} V_{11}^L & \\ & V_{12}^L \\ V_{21}^L & \\ & V_{22}^L \end{pmatrix},
\end{aligned}
\tag{14}
$$

as in (12).

Suppose 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 partitioned into leaves $\{r_i\}_{1\leq i\leq m}$ and $\{c_i\}_{1\leq i\leq m}$ as in (8). By the *rID*s and *cID*s applied in the construction of (13), we have obtained skeleton index sets $\hat{r}_i \subset r_i$ and $\hat{c}_i \subset c_i$. Then

$$
S_{ij}^L = \begin{pmatrix} A_{\hat{r}_{(i-1)m/2+1},\hat{c}_{(j-1)m/2+1}} & \cdots & A_{\hat{r}_{(i-1)m/2+1},\hat{c}_{jm/2}} \\ \vdots & \ddots & \vdots \\ A_{\hat{r}_{im/2},\hat{c}_{(j-1)m/2+1}} & \cdots & A_{\hat{r}_{im/2},\hat{c}_{jm/2}} \end{pmatrix} \tag{15}
$$

for $i, j = 1, 2$.

As explained in Section 2.2, each non-zero block S_{ij}^L in S^L is a submatrix of A_{ij} consisting of important rows and columns of A_{ij} for $i, j = 1, 2$. Hence, S_{ij}^L inherits the complementary low-rank property of A_{ij} and is a complementary low-rank matrix. Suppose $T_{X,ij}$ and $T_{\Omega,ij}$ are the dyadic trees of the row and column spaces of A_{ij} with $m/2$ leafs and $L-1$ depth, then according to Section 2.2, S_{ij}^L has compressible row and column dyadic trees $\hat{T}_{X,ij}$ and $\hat{T}_{\Omega,ij}$ with $m/4$ leafs and $L-2$ depth.

Next, we apply MSCS to each S_{ij}^L in a recursive way. In particular, we divide each S_{ij}^L into a 2×2 block matrix according to the nodes at the second level of its row and column dyadic trees:

$$
S_{ij}^L = \begin{pmatrix} (S_{ij}^L)_{11} & (S_{ij}^L)_{12} \\ (S_{ij}^L)_{21} & (S_{ij}^L)_{22} \end{pmatrix}. \tag{16}
$$

After constructing the LRCS of the (k, ℓ)-th block of S_{ij}^L, i.e., $(S_{ij}^L)_{k\ell} = (U_{ij}^{L-1})_{k\ell}(S_{ij}^{L-1})_{k\ell}(V_{ij}^{L-1})_{k\ell}$ for $k, \ell = 1, 2$, we assemble them to obtain the MSCS of S_{ij}^L as follows:

$$
S_{ij}^L = U_{ij}^{L-1} S_{ij}^{L-1} V_{ij}^{L-1}, \tag{17}
$$

where

$$
\begin{aligned}
U_{ij}^{L-1} &= \begin{pmatrix} (U_{ij}^{L-1})_{11} & & (U_{ij}^{L-1})_{12} & \\ & (U_{ij}^{L-1})_{21} & & (U_{ij}^{L-1})_{22} \end{pmatrix}, \\
S_{ij}^{L-1} &= \begin{pmatrix} (S_{ij}^{L-1})_{11} & & & \\ & & (S_{ij}^{L-1})_{21} & \\ & (S_{ij}^{L-1})_{12} & & \\ & & & (S_{ij}^{L-1})_{22} \end{pmatrix}, \\
V_{ij}^{L-1} &= \begin{pmatrix} (V_{ij}^{L-1})_{11} & \\ & (V_{ij}^{L-1})_{12} \\ (V_{ij}^{L-1})_{21} & \\ & (V_{ij}^{L-1})_{22} \end{pmatrix},
\end{aligned}
\tag{18}
$$

according to Section 2.3.

Finally, we organize the factorizations in (17) for all $i, j = 1, 2$ to form a factorization of S^L as

$$
S^L \approx U^{L-1} S^{L-1} V^{L-1}, \tag{19}
$$

where

$$
\begin{aligned}
U^{L-1} &= \begin{pmatrix} U_{11}^{L-1} & & & \\ & U_{21}^{L-1} & & \\ & & U_{12}^{L-1} & \\ & & & U_{11}^{L-1} \end{pmatrix}, \\
S^{L-1} &= \begin{pmatrix} S_{11}^{L-1} & & & \\ & & S_{21}^{L-1} & \\ & S_{12}^{L-1} & & \\ & & & S_{22}^{L-1} \end{pmatrix}, \\
V^{L-1} &= \begin{pmatrix} V_{11}^{L-1} & & & \\ & V_{12}^{L-1} & & \\ & & V_{21}^{L-1} & \\ & & & S_{22}^{L-1} \end{pmatrix},
\end{aligned}
\tag{20}
$$

leading to a second level factorization of A:

$$
A \approx U^L U^{L-1} S^{L-1} V^{L-1} V^L.
$$

Figure 8 visualizes the recursive MSCS of S^L in (19) when A is a complementary low-rank matrix with dyadic trees of depth 5 and 16 leaf nodes in Figure 2.

Comparing (13), (14), (19), and (20), we can see a fractal structure in each level of the middle factor S^ℓ for $\ell = L$ and $L-1$. For example in (20) (see Figure 8 for its visulaization), S^{L-1} has 4 submatrices S_{ij}^{L-1} with the same structure as S^L for all i and j. S_{ij}^{L-1} can be factorized into a product of three matrices with the same sparsity structure as the factorization $S^L = U^{L-1} S^{L-1} V^{L-1}$. Hence, we can apply MSCS recursively to each S^ℓ and assemble matrix factors hierarchically for $\ell = L, \ldots, L/2$ to obtain

$$
A \approx U^L U^{L-1} \cdots U^h S^h V^h \cdots V^{L-1} V^L, \tag{21}
$$

where $h = L/2$. In the ℓ-th recursive MSCS, S^ℓ has $2^{2(L-\ell+1)}$ dense submatrices with compressible row and column dyadic trees with $\frac{m}{2^{2(L-\ell+1)}}$ leaves and depth $L - 2(L-\ell+1)$. Hence, the recursive MSCS stops after $h = L/2$ iterations when S^h doesn't contain compressible submatrices.

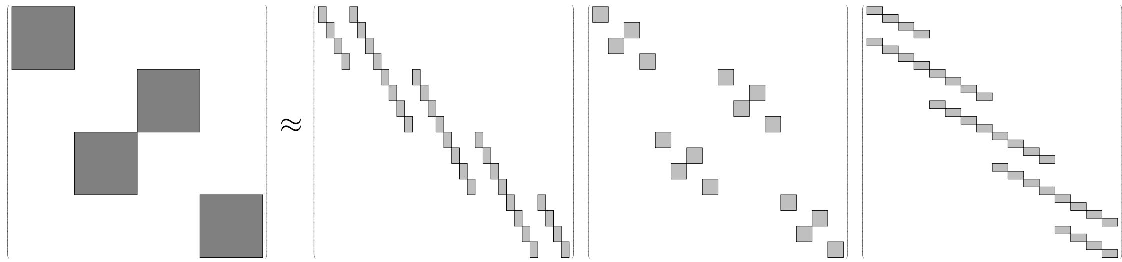


Figure 8: The visualization of the recursive MSCS of $S^L = U^{L-1} S^{L-1} V^{L-1}$ in (19) when A is a complementary low-rank matrix with dyadic trees of depth 5 and 16 leaf nodes in Figure 2.

3 Numerical results

This section presents several numerical examples to demonstrate the effectiveness of the algorithms proposed above. The first three examples are complementary low-rank matrices coming from non-uniform Fourier transform, Fourier integral operators, and special function transforms. The last two examples are hierarchical complementary matrices [31] from 2D Helmholtz boundary integral methods in the high-frequency regime. IDBF is able to reduce the construction time of the data-sparse representation of the HSS-type complementary matrix in [31] from $N^{1.5}$ to nearly linear scaling. IDBF can also accelerate the factorization time of the hierarchical complementary matrix in [32] to nearly linear scaling for 3D high-frequency boundary integral methods. After factorization, the application time of matrices in these two integral methods is nearly linear scaling. We leave the trivial extension to 3D high-frequency integral methods to the reader and only present the results for 2D problems. All implementations are in MATLAB® on a server computer with a single thread and 3.2 GHz CPU. This new framework will be incorperated into the ButterflyLab[2] in the future.

Let $\{u^d(x), x \in X\}$ and $\{u^a(x), x \in X\}$ denote the results given by the direct matrix-vector multiplication and the butterfly factorization. The accuracy of applying the butterfly factorization algorithm is estimated by the following relative error

$$\epsilon^a = \sqrt{\frac{\sum_{x\in S} |u^a(x) - u^d(x)|^2}{\sum_{x\in S} |u^d(x)|^2}}, \tag{22}$$

where S is a point set of size 256 randomly sampled from X.

In all of our examples, we use Mock-Chebyshev grid points and the oversampling parameter p in the linear scaling ID is set to 1. The number of Mock-Chebyshev grid points is also called the truncation rank (the rank parameter k) in IDs.

Example 1. Our first example is to evaluate a one-dimensional FIO of the following form:

$$u(x) = \int_{\mathbb{R}} e^{2\pi\imath\Phi(x,\xi)} \hat{f}(\xi) d\xi, \tag{23}$$

where $\hat{f}$ is the Fourier transform of f, and $\Phi(x, \xi)$ is a phase function given by

$$\Phi(x, \xi) = x \cdot \xi + c(x)|\xi|, \quad c(x) = (2 + 0.2\sin(2\pi x))/16. \tag{24}$$

The discretization of (23) is

$$u(x_i) = \sum_{\xi_j} e^{2\pi\imath\Phi(x_i,\xi_j)} \hat{f}(\xi_j), \quad i, j = 1, 2, \ldots, N, \tag{25}$$

[2]Available on `https://github.com/ButterflyLab`.

where $\{x_i\}$ and $\{\xi_j\}$ are uniformly distributed points in $[0, 1)$ and $[-N/2, N/2)$ following

$$x_i = (i-1)/N \text{ and } \xi_j = j - 1 - N/2. \tag{26}$$

(25) can be represented in a matrix form as $u = Kg$, where $u_i = u(x_i)$, $K_{ij} = e^{2\pi\imath\Phi(x_i,\xi_j)}$ and $g_j = \hat{f}(\xi_j)$. The matrix K satisfies the complementary low-rank property with a rank parameter k independent of the problem size N when ξ is sufficiently far away from the origin as proved in [36, 43]. To make the presentation simpler, we will directly apply IDBF to the whole K instead of performing a polar transform as in [36] or apply IDBF hierarchically as in [44]. Hence, due to the non-smoothness of the $\Phi(x, \xi)$ at $\xi = 0$, submatrices intersecting with or close to the line $\xi = 0$ have a local rank increasing slightly in N, while other submatrices have rank independent of N. Figure 9 summarizes the results of this example for different grid sizes N with the same truncation rank $r = 30$ and tolerance $tol = 10^{-8}$. We see that the IDBF applied to the whole matrix K has $O(N \log^2(N))$ factorization and application time. The running time agrees with the scaling of the number of non-zero entries required in the data-sparse representation. In fact, when N is large enough, the number of non-zero entries in the IDBF tends to scale as $O(N \log N)$, which means that the numerical scaling can approach to $O(N \log N)$ in both factorization and application when N is large enough.

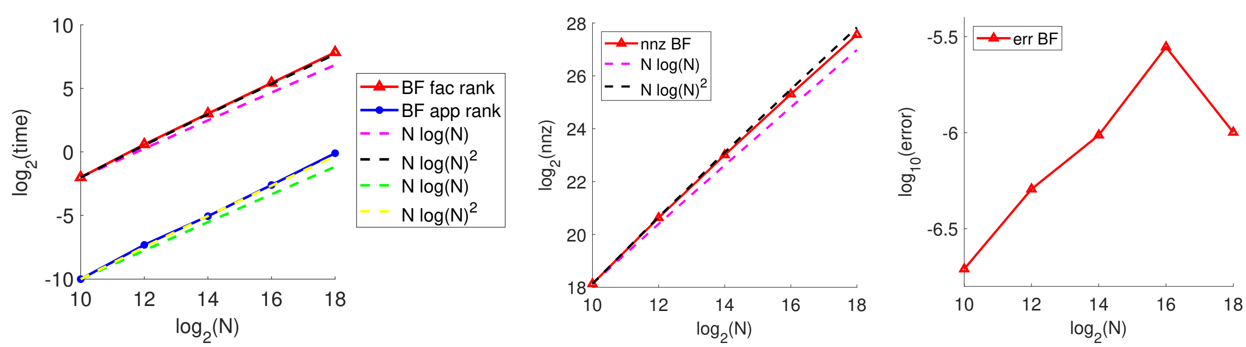


Figure 9: Numerical results for the FIO given in (25). N is the size of the matrix; nnz is the number of non-zero entries in the butterfly factorization, err is the approximation error of the IDBF matvec.

Example 2. Next, we provide an example of a special function transform, the evaluation of Schlömilch expansions [45] at $g_k = \frac{k-1}{N}$ for $1 \leq k \leq N$:

$$u_k = \sum_{n=1}^{N} c_n J_\nu(g_k \omega_n), \tag{27}$$

where J_ν is the Bessel function of the first kind with parameter $\nu = 0$, and $\omega_n = n\pi$. It is demonstrated in [14] that (27) can be represented via a matvec $u = Kg$, where K satisfies the complementary low-rank property. An arbitrary entry of K can be calculated in $O(1)$ operations [46] and hence IDBF is suitable for accelerating the matvec $u = Kg$. Other similar examples can be found in [45] and they can be also evaluated by IDBF with the same operation counts. Figure 10 summarizes the results of this example for different problem sizes N with the same truncation rank $r = 30$ and tolerance $tol = 10^{-8}$. The results show that IDBF applied to this example has $O(N\log^2(N))$ factorization and application time. The running time agrees with the scaling of the number of non-zero entries required in the data-sparse representation to guarantee the approximation accuracy. In fact, when N is large enough, the number of non-zero entries in the IDBF tends to scale as $O(N\log N)$, which means that the numerical scaling can approach to $O(N\log N)$ in both factorization and application when N is large enough.

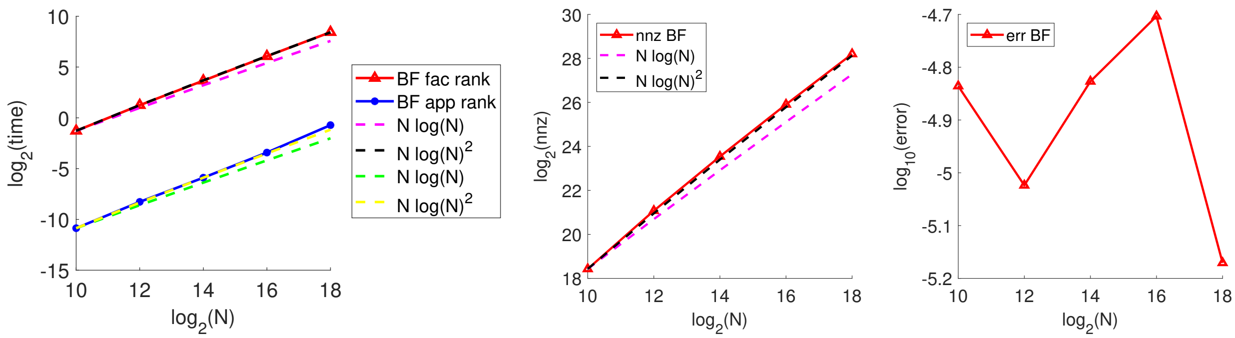


Figure 10: Numerical results for the Schlömilch expansions given in (27). N is the size of the matrix; nnz is the number of non-zero entries in the butterfly factorization, err is the approximation error of the IDBF matvec.

Example 3. In this example, we consider the one-dimensional non-uniform Fourier transform as follows:

$$u_k = \sum_{n=1}^{N} e^{-2\pi\imath x_n \omega_k} g_n, \tag{28}$$

for $1 \le k \le N$, where x_n is randomly selected in $[0, 1)$, and ω_k is randomly selected in $[-\frac{N}{2}, \frac{N}{2})$ according to uniform distributions in these intervals.

Figure 11 summarizes the results of this example for different grid sizes N with the same truncation rank $r = 30$ and tolerance $tol = 10^{-8}$. Numerical results show that IDBF admits at most $O(N\log^2(N))$ factorization and application time for the non-uniform Fourier transform. The running time agrees with the scaling of the number of non-zero entries required in the data-sparse representation. In fact, when N is large enough, the number of non-zero entries in the IDBF tends to scale as $O(N\log N)$, which means that the numerical scaling can approach to $O(N\log N)$ in both factorization and application when N is large enough.

Example 4. The fourth example is from electric field integral equations (EFIEs). In EFIEs, the linear system to be solved using the method of moments is of the following form [13]

$$Zx = b,$$

where Z is an impedance matrix with

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

where $e \approx 2.718$, $\gamma \approx 1.781$, μ_o is the free-space permeability, $k = 2\pi/\lambda_o$ is the wavenumber, λ_o represents the free-space wavelength, $H_0^{(2)}$ denotes the zeroth-order Hankel function of the second

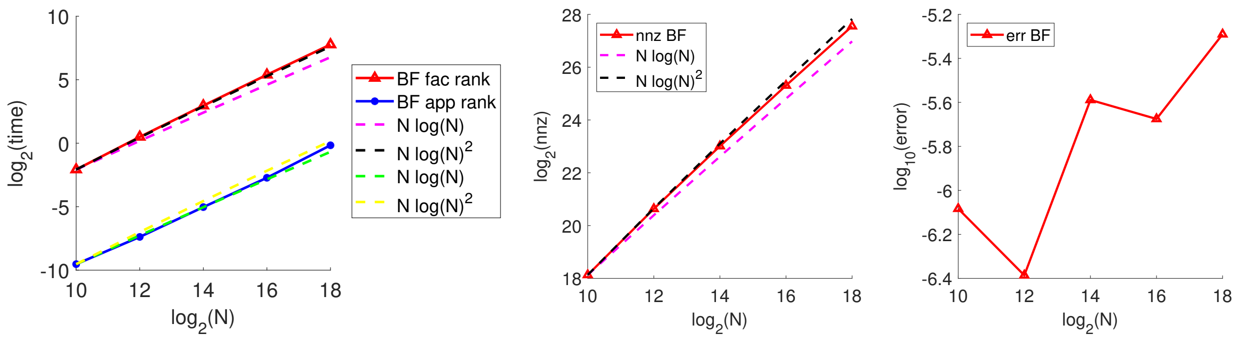
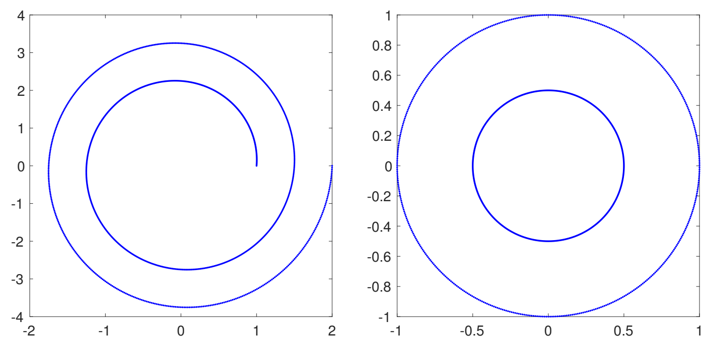


Figure 11: Numerical results for the NUFFT given in (28). N is the size of the matrix; nnz is the number of non-zero entries in the butterfly factorization, err is the approximation error of the IDBF matvec.



(a) (b)

Figure 12: The two scatterers used in Example 4 and 5: (a) a spiral object; (b) a round object with a hole in center which is the port.

kind, ω_i is the length of the i-th segment of the scatter object, ρ_i is the center of the i-th segment, ω is a constant of order k.

It was shown in [13, 31] that the impedance matrix of the EFIE by the method of moments for analyzing scattering from two-dimensional objects admits a HSS-type complementary low-rank property, i.e., off-diagonal blocks are complementary low-rank matrices. The method in [31] requires $O(N^{1.5}\log N)$ operations to compress the impedance matrix via a slower version of butterfly factorization. After compression, it requires $O(N\log^2(N))$ operations to apply the impedance matrix and makes it possible to design efficient iterative solvers to solve the linear system for the impedance matrix. Replacing the butterfly factorization in [31] with IDBF, we reduce the factorization time to $O(N\log^2(N))$ as well.

Figure 13 shows the results of the fast matvec of the impedance matrix from an 2D EFIE generated with a spiral object as shown in Figure 12 (a). We vary the number of discretization segments N of the scatter object and let $k = O(N)$ in the construction of Z. In the IDBF, we use the same truncation rank $r = 40$ and tolerance $tol = 10^{-4}$. Numerical results verifies the $O(N\log^2(N))$ scaling for both the factorization and application of the new HSS-type butterfly factorization by IDBF.

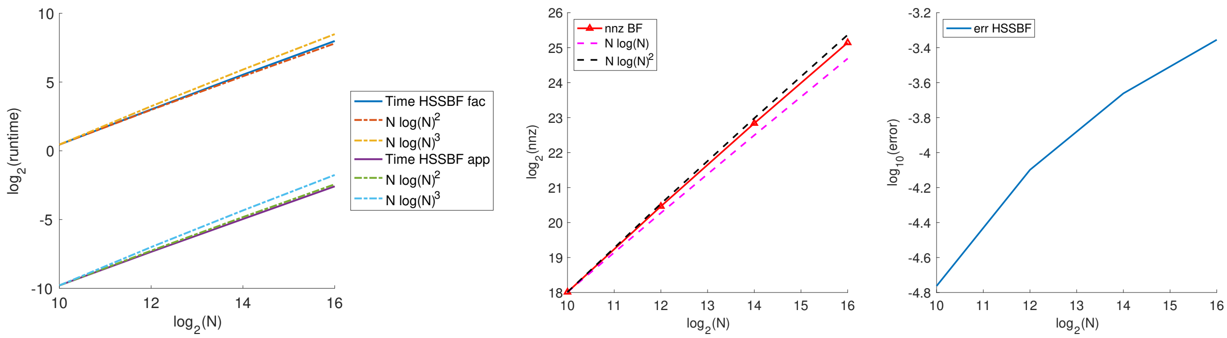


Figure 13: Numerical results for the 2D electric field integral equation. N is the size of the matrix; nnz is the number of non-zero entries in the butterfly factorization, err is the approximation error of the matvec by hierarchically applying IDBF.

Example 5. The fifth example is from combined field integral equations (CFIEs). Similar to the ideas in [13, 31] for EFIE, we show that the impedance matrix of the CFIE[3] by the method of moments for analyzing scattering from two-dimensional objects also admits a HSS-type complementary low-rank property. Applying the same HSS-type butterfly factorization by IDBF, we obtain $O(N \log^2(N))$ scaling for both the factorization and application time for impedance matrices of CFIEs. This makes it possible to design efficient iterative solvers to solve the linear system for the impedance matrix. Figure 14 shows the results of the fast matvec of the impedance matrix from an 2D CFIE generated with a round object as shown in Figure 12 (b). We vary grid sizes N with the same truncation rank $r = 40$ and tolerance $tol = 10^{-4}$. Numerical results verifies the $O(N \log^2(N))$ scaling for both the factorization and application of the new HSS-type butterfly factorization by IDBF.

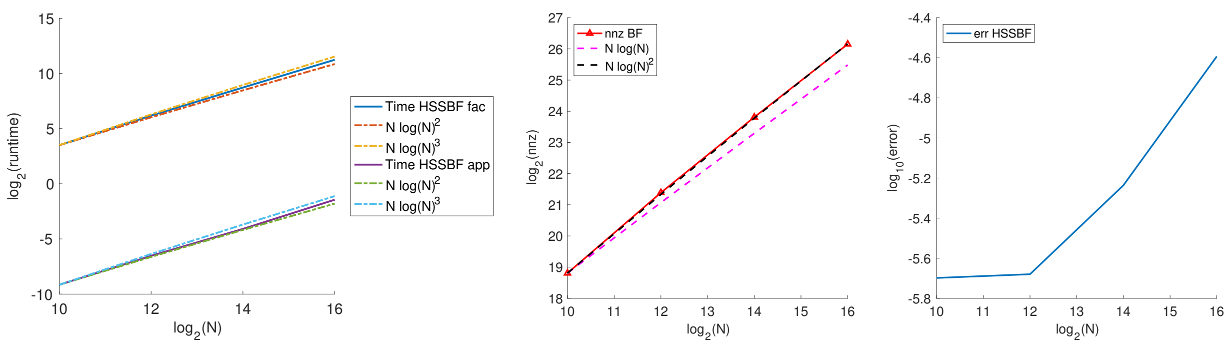


Figure 14: Numerical results for the 2D combined field integral equation. N is the size of the matrix; nnz is the number of non-zero entries in the butterfly factorization, err is the approximation error of the matvec by hierarchically applying IDBF.

[3]Codes for generating the impedance matrix are from a MATLAB package "emsolver" available at `https://github.com/dsmi/emsolver`.

4 Conclusion and discussion

This paper introduces an interpolative decomposition butterfly factorization as a data-sparse approximation of complementary low-rank matrices. It represents such an $N \times N$ dense matrix as a product of $O(\log N)$ sparse matrices. The factorization and application time, and the memory of IDBF all scale as $O(N \log N)$. IDBF can also be applied to the HSS-type complementary low-rank matrices introduced in [13, 31]. The new HSS-type butterfly factorization based on IDBF also admits nearly linear scaling for both factorization and application.

Acknowledgments. H. Yang thanks the support of the start-up grant by the Department of Mathematics at the National University of Singapore.

References

[1] N. Halko, P. Martinsson, and J. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

[2] Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert. A fast randomized algorithm for the approximation of matrices. *Applied and Computational Harmonic Analysis*, 25(3):335 – 366, 2008.

[3] H. Cheng, Z. Gimbutas, P. Martinsson, and V. Rokhlin. On the compression of low rank matrices. *SIAM Journal on Scientific Computing*, 26(4):1389–1404, 2005.

[4] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[5] Michael W Mahoney and Petros Drineas. CUR matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, 2009.

[6] Wolfgang Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. I. Introduction to $\mathcal{H}$-matrices. *Computing*, 62(2):89–108, 1999.

[7] 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.

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

[9] W. Hackbusch and S. Börm. Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices. *Computing*, 69(1):1–35, 2002.

[10] W. Hackbusch, B. Khoromskij, and S. A. Sauter. On h2-matrices. In Hans-Joachim Bungartz, Ronald H. W. Hoppe, and Christoph Zenger, editors, *Lectures on Applied Mathematics*, pages 9–29, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[11] Jianlin Xia, Shivkumar Chandrasekaran, Ming Gu, and Xiaoye S. Li. Fast algorithms for hierarchically semiseparable matrices. *Numerical Linear Algebra with Applications*, 17(6):953–976, 2010.

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

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

[14] 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.

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

[16] Yingzhou Li and Haizhao Yang. Interpolative butterfly factorization. *SIAM Journal on Scientific Computing*, 39(2):A503–A531, 2017.

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

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

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

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

[21] 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.

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

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

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

[25] J. 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.

[26] V. Minden, K. Ho, A. Damle, and L. Ying. A recursive skeletonization factorization based on strong admissibility. *Multiscale Modeling & Simulation*, 15(2):768–796, 2017.

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

[28] Bjrn 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.

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

[30] 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.

[31] Y. Liu, H. Guo, and E. 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.

[32] H. Guo, Y. Liu, J. Hu, and E. 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.

[33] D. S. Seljebotn. Wavemoth-fast spherical harmonic transforms by butterfly matrix compression. *The Astrophysical Journal Supplement Series*, 199(1):5, 2012.

[34] Mark Tygert. Fast algorithms for spherical harmonic expansions, {III}. *Journal of Computational Physics*, 229(18):6181 – 6192, 2010.

[35] James Bremer and Haizhao Yang. Fast algorithms for jacobi expansions via nonoscillatory phase functions. *arXiv:1803.03889 [math.NA]*, 2018.

[36] 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.

[37] Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. Randomized algorithms for the low-rank approximation of matrices. *Proc. Natl. Acad. Sci. USA*, 104(51):20167–20172, 2007.

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

[39] J. Chiu and L. Demanet. Sublinear randomized algorithms for skeleton decompositions. *SIAM Journal on Matrix Analysis and Applications*, 34(3):1361–1383, 2013.

[40] R. Platte, L. Trefethen, and A. Kuijlaars. Impossibility of fast stable approximation of analytic functions from equispaced samples. *SIAM Review*, 53(2):308–318, 2011.

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

[42] 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.

[43] Yingzhou Li, Haizhao Yang, and Lexing Ying. A multiscale butterfly aglorithm for Fourier integral operators. *Multiscale Modeling and Simulation*, to appear.

[44] Y. Li, H. Yang, and L. Ying. A multiscale butterfly algorithm for multidimensional fourier integral operators. *Multiscale Modeling & Simulation*, 13(2):614–631, 2015.

[45] A. Townsend. A fast analysis-based discrete hankel transform using asymptotic expansions. *SIAM Journal on Numerical Analysis*, 53(4):1897–1917, 2015.

[46] James Bremer. An algorithm for the rapid numerical evaluation of bessel functions of real orders and arguments. *Advances in Computational Mathematics*, to appear.