Letter to the Editor

Multidimensional butterfly factorization

CrossMark

Yingzhou Li [b,*], Haizhao Yang [c], Lexing Ying [a,b]

[a] *Department of Mathematics, Stanford University, United States*
[b] *ICME, Stanford University, United States*
[c] *Department of Mathematics, Duke University, United States*

A R T I C L E   I N F O

A B S T R A C T

This paper introduces the multidimensional butterfly factorization as a data-sparse representation of multidimensional kernel matrices that satisfy the complementary low-rank property. This factorization approximates such a kernel matrix of size $N \times N$ with a product of $O(\log N)$ sparse matrices, each of which contains $O(N)$ nonzero entries. We also propose efficient algorithms for constructing this factorization when either (i) a fast algorithm for applying the kernel matrix and its adjoint is available or (ii) every entry of the kernel matrix can be evaluated in $O(1)$ operations. For the kernel matrices of multidimensional Fourier integral operators, for which the complementary low-rank property is not satisfied due to a singularity at the origin, we extend this factorization by combining it with either a polar coordinate transformation or a multiscale decomposition of the integration domain to overcome the singularity. Numerical results are provided to demonstrate the efficiency of the proposed algorithms.

© 2017 Elsevier Inc. All rights reserved.

1. Introduction

1.1. Problem statement

This paper is concerned with the efficient evaluation of

$$u(x) = \sum_{\xi \in \Omega} K(x, \xi) g(\xi), \quad x \in X, \tag{1}$$

where X and Ω are typically point sets in $\mathbb{R}^d$ for $d \geq 2$, $K(x, \xi)$ is a kernel function that satisfies a complementary low-rank property, $g(\xi)$ is an input function for $\xi \in \Omega$, and $u(x)$ is an output function

---

* Corresponding author.
  E-mail address: ryanli@stanford.edu (Y. Li).

for $x \in X$. To define this complementary low-rank property for multidimensional kernel matrices, we first assume that without loss of generality there are N points in each point set. In addition, the domains X and Ω are associated with two hierarchical trees T_X and T_Ω, respectively, where each node of these trees represents a subdomain of X or Ω. Both T_X and T_Ω are assumed to have $L = O(\log N)$ levels with X and Ω being the roots at level 0. The computation of (1) is essentially a matrix–vector multiplication

$$u = Kg,$$

where $K := (K(x, \xi))_{x \in X, \xi \in \Omega}$, $g := (g(\xi))_{\xi \in \Omega}$, and $u := (u(x))_{x \in X}$ by a slight abuse of notations. The matrix K is said to satisfy the *complementary low-rank property* if for any level ℓ between 0 and L and for any node A on the ℓ-th level of T_X and any node B on the $(L - \ell)$-th level of T_Ω, the submatrix $K_{A,B} := (K(x_i, \xi_j))_{x_i \in A, \xi_j \in B}$ is numerically low-rank with the rank bounded by a uniform constant independent of N. In most applications, this numerical rank is bounded polynomially in $\log(1/\epsilon)$ for a given precision ϵ. A well-known example of such a matrix is the multidimensional Fourier transform matrix.

For a complementary low-rank kernel matrix K, the *butterfly algorithm* developed in [1,2,13,14,16] enables one to evaluate the matrix–vector multiplication in $O(N \log N)$ operations. More recently in [7], we introduced the *butterfly factorization* as a data-sparse multiplicative factorization of the kernel matrix K in the one-dimensional case $(d = 1)$:

$$K \approx U^L G^{L-1} \cdots G^{L/2} M^{L/2} \left(H^{L/2}\right)^* \cdots \left(H^{L-1}\right)^* \left(V^L\right)^*, \tag{2}$$

where the depth $L = O(\log N)$ is assumed to be an even number and every factor in (2) is a sparse matrix with $O(N)$ nonzero entries. Here the superscript of a matrix denotes the level of the factor rather than the power of a matrix. This factorization requires $O(N \log N)$ memory and applying (2) to any vector takes $O(N \log N)$ operations once the factorization is computed. In fact, one can view the factorization in (2) as a compact algebraic representation of the butterfly algorithm. In [7], we also introduced algorithms for constructing the butterfly factorization for the following two cases:

(i) A black-box routine for rapidly computing Kg and K^*g in $O(N \log N)$ operations is available;
(ii) A routine for evaluating any entry of K in $O(1)$ operations is given.

In this paper, we turn to the butterfly factorization for the multidimensional problems and describe how to construct them for these two cases.

When the kernel strictly satisfies the complementary low-rank property (e.g., the non-uniform FFT), the algorithms proposed in [7] can be generalized in a rather straightforward way. This is presented in detail in Section 2.

However, many important multidimensional kernel matrices fail to satisfy the complementary low-rank property in the entire domain $X \times \Omega$. Among them, the most significant example is probably the Fourier integral operator, which typically has a singularity at the origin $\xi = 0$ in the Ω domain. For such an example, existing butterfly algorithms provide two solutions:

- The first one, proposed in [2], removes the singularity by applying a polar transformation that maps the domain Ω into a new domain P. After this transformation, the new kernel matrix defined on $X \times P$ satisfies the complementary low-rank property and one can then apply the butterfly factorization in the X and P domain instead. This is discussed in detail in Section 3 and we refer to this algorithm as the *polar butterfly factorization* (PBF).
- The second solution proposed in [8] is based on the observation that, though not on the entire Ω domain, the complementary low-rank property holds in subdomains of Ω that are well separated from the origin

in a certain sense. For example, one can start by partitioning the domain Ω into a disjoint union of a small square Ω_C covering $\xi = 0$ and a sequence of dyadic coronas Ω_t, i.e., $\Omega = \Omega_C \cup (\cup_t \Omega_t)$. Accordingly, one can rewrite the kernel evaluation (1) as a summation of the form

$$K = K_C R_C + \sum_t K_t R_t, \tag{3}$$

where K_C and K_t are the kernel matrices restricted to $X \times \Omega_C$ and $X \times \Omega_t$, R_C and R_t are the operators of restricting the input functions defined on Ω to the subdomain Ω_C and Ω_t, respectively. In fact, each kernel K_t satisfies the complementary low-rank property and hence one can approximate it with the multidimensional butterfly factorization in Section 2. Combining the factorizations for all K_t with (3) results the *multiscale butterfly factorization* (MBF) for the entire matrix K and this will be discussed in detail in Section 4.

In order to simplify the presentation, this paper focuses on the two dimensional case ($d = 2$). Furthermore, we assume that the points in X and Ω are uniformly distributed in both domains as follows:

$$X = \left\{ x = \left(\frac{n_1}{n}, \frac{n_2}{n} \right), 0 \le n_1, n_2 < n \text{ with } n_1, n_2 \in \mathbb{Z} \right\} \tag{4}$$

and

$$\Omega = \left\{ \xi = (n_1, n_2), -\frac{n}{2} \le n_1, n_2 < \frac{n}{2} \text{ with } n_1, n_2 \in \mathbb{Z} \right\}, \tag{5}$$

where n is the number of points in each dimension and $N = n^2$. This is the standard setup for two dimensional Fourier transforms and FIOs.

1.2. Related work

For a complementary low-rank kernel matrix K, the butterfly algorithm provides an efficient way for evaluating (1). It was initially proposed in [13] and further developed in [2,6,8,14–17,20]. One can roughly classify the existing butterfly algorithms into two groups:

- The first group (e.g. [14,16,17]) requires a precomputation stage for constructing the low-rank approximations of the numerically low-rank submatrices of (1). This precomputation stage typically takes $O(N^2)$ operations and uses $O(N \log N)$ memory. Once the precomputation is done, the evaluation of (1) can be carried out in $O(N \log N)$ operations.
- The second group (e.g. [2,6,8,15]) assumes prior knowledge of analytic properties of the kernel function. Under such analytic assumptions, one avoids precomputation by writing down the low-rank approximations for the numerically low-rank submatrices explicitly. These algorithms typically evaluate (1) with $O(N \log N)$ operations.

In a certain sense, the algorithms proposed in this paper can be viewed as a compromise of these two types. On the one hand, it makes rather weak assumptions about the kernel. Instead of requiring the kernel function as was done for the second type, we only assume that either (i) a fast matrix–vector multiplication routine or (ii) a kernel matrix sampling routine is available. On the other hand, these new algorithms reduce the precomputation cost to $O(N^{3/2} \log N)$, as compared to the quadratic complexity of the first group.

The multidimensional butterfly factorization can also be viewed as a process of recovering a structured matrix via either sampling or matrix–vector multiplication. There has been a sequence of articles in this line of research. For example, we refer to [4,9,18] for recovering numerically low-rank matrices, [11] for recovering

an HSS matrices, and [10] for recovering $\mathcal{H}$-matrices. This paper generalizes the work of [7] by considering complementary low-rank matrices coming from multidimensional problems.

1.3. Organization

The rest of this paper is organized as follows. Section 2 reviews the basic tools and describes the *multidimensional butterfly factorization* for kernel matrices that strictly satisfy the complementary low-rank property. We then extend it in two different ways to address the multidimensional Fourier integral operators. Section 3 introduces the polar butterfly factorization based on the polar butterfly algorithm proposed in [2]. Section 4 discusses the multiscale butterfly factorization based on the multiscale butterfly algorithm proposed in [8]. Finally, in Section 5, we conclude with some discussions.

2. Two-dimensional butterfly factorization

This section presents the two-dimensional butterfly factorization for a kernel matrix $K = (K(x, \xi))_{x \in X, \xi \in \Omega}$ that satisfies the complementary low-rank property in $X \times \Omega$ with X and Ω given in (4) and (5).

2.1. Randomized low-rank factorization

The butterfly factorization relies heavily on randomized procedures for computing low-rank factorizations. For a matrix $Z \in \mathbb{C}^{m \times n}$, a rank-$r$ approximation in 2-norm can be computed via the truncated singular value decomposition (SVD),

$$Z \approx U_0 \Sigma_0 V_0^*, \tag{6}$$

where $U_0 \in \mathbb{C}^{m \times r}$ and $V_0 \in \mathbb{C}^{n \times r}$ are unitary matrices, $\Sigma_0 \in \mathbb{R}^{r \times r}$ is a diagonal matrix with the largest r singular values of Z in decreasing order.

Once $Z \approx U_0 \Sigma_0 V_0^*$ is available, we can also construct different low-rank factorizations of Z in three forms:

$$Z \approx USV^*, \quad U = U_0 \Sigma_0, \quad S = \Sigma_0^{-1}, \quad V^* = \Sigma_0 V_0^*; \tag{7}$$

$$Z \approx UV^*, \quad U = U_0 \Sigma_0, \quad V^* = V_0^*; \tag{8}$$

$$Z \approx UV^*, \quad U = U_0, \quad V^* = \Sigma_0 V_0^*. \tag{9}$$

As we shall see, the butterfly factorization uses each of these three forms in different stages of the algorithm.

In [7], we showed that the rank-r SVD (6) can be constructed approximately via either random matrix–vector multiplication [4,12] or random sampling [3,18,19]. In both cases, the key is to find accurate approximate bases for both the column and row spaces of Z and approximate the largest r singular values using these bases.

SVD via random matrix–vector multiplication. This algorithm proceeds as follows:

- This algorithm first applies Z to a Gaussian random matrix $C \in \mathbb{C}^{n \times (r+k)}$ and its adjoint Z^* to a Gaussian random matrix $R \in \mathbb{C}^{m \times (r+k)}$, where k is the oversampling constant.
- Second, computing the pivoted QR decompositions of ZC and Z^*R identifies unitary matrices $Q_{col} \in \mathbb{C}^{m \times r}$ and $Q_{row} \in \mathbb{C}^{n \times r}$, which approximately span the column and row spaces of Z, respectively.

- Next, the algorithms seeks a matrix M that satisfies

$$Z \approx Q_{col} M Q_{row}^*$$

by setting $M = (R^* Q_{col})^\dagger R^* Z C (Q_{row}^* C)^\dagger$, where $(\cdot)^\dagger$ denotes the pseudo inverse.
- Finally, combining the singular value decomposition $M = U_M \Sigma_M V_M^*$ of the matrix M with the above approximation results in the desired approximate rank-r SVD

$$Z \approx (Q_{col} U_M) \Sigma_M (Q_{row} V_M)^*.$$

Suppose that the cost of applying Z and Z^* to an arbitrary vector is $C_Z(m, n)$. Then the construction complexity of this procedure is $O(C_Z(m, n)r + \max(m, n)r^2)$. As we shall see later, when the black-box routines for rapidly applying K and K^* are available, this procedure would be embedded into the algorithms for constructing the butterfly factorizations.

SVD via random sampling. The main steps and the intuition behind this algorithm are given below. The reader is referred to [3,18,19] for a detailed algorithm description.

- The first stage discovers the representative columns and rows progressively via computing multiple pivoted QR factorizations on randomly selected rows and columns of Z. The representative columns and rows are set to be empty initially. As the procedure processes, more and more columns (rows) are marked as *representative* and they are used in turn to discover new representative rows (columns). The procedure stops when the sets of the representative rows and columns stabilize. At this point, the representative columns (rows) approximately span the column (row) spaces of Z.
- Second, computing the pivoted QR decompositions of the representative columns and rows identifies unitary matrices $Q_{col} \in \mathbb{C}^{m \times r}$ and $Q_{row} \in \mathbb{C}^{n \times r}$, which approximately span the column and row spaces of Z, respectively.
- Next, the algorithm seeks a matrix M that satisfies

$$Z \approx Q_{col} M Q_{row}^*.$$

This is done by restricting this equation to a random row set I_{row} and a random column set I_{col} and consider

$$Z(I_{row}, I_{col}) \approx Q_{col}(I_{row}, :) M Q_{row}(I_{col}, :)^*.$$

Here both I_{row} and I_{col} are of size $O(r)$ and we require I_{row} and I_{col} to contain the set of representative rows and columns, respectively. From the above equation, we can solve M by setting

$$M = (Q_{col}(I_{row}, :))^\dagger Z(I_{row}, I_{col})(Q_{row}(I_{col}, :)^*)^\dagger.$$

- Finally, combining the singular value decomposition $M = U_M \Sigma_M V_M^*$ of the matrix M with the approximation $Z \approx Q_{col} M Q_{row}^*$ results in the desired approximate rank-r SVD

$$Z \approx (Q_{col} U_M) \Sigma_M (Q_{row} V_M)^*.$$

The construction complexity of this procedure is $O(\max(m, n)r^2)$ in practice. When an arbitrary entry of Z can be evaluated in $O(1)$ operations, this procedure is the method of choice for constructing low-rank factorizations.
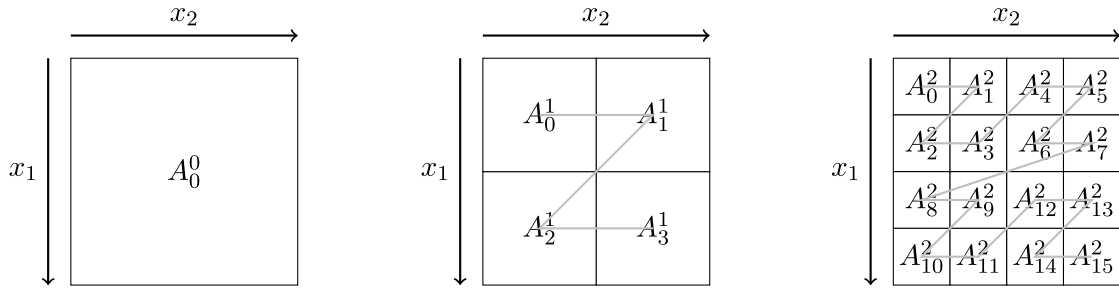
Fig. 1. An illustration of Z-order curve cross levels. The superscripts indicate the different levels while the subscripts indicate the index in the Z-ordering. The light gray lines show the ordering among the subdomains on the same level. Left: The root at level 0. Middle: At level 1, the domain A_0^0 is divided into 2×2 subdomains A_i^1 with $i \in \mathcal{I}^1 = \{0, 1, 2, 3\}$. These 4 subdomains are ordered according to the Z-ordering. Right: At level 2, the domain A_0^0 is divided into 4×4 subdomains A_i^2 with $i \in \mathcal{I}^2 = \{0, 1, \ldots, 15\}$. These 16 subdomains are ordered similarly.

2.2. Notations and overall structure

We adopt the notation of the one-dimensional butterfly factorization introduced in [7] and adjust them to the two-dimensional case of this paper.

Recall that n is the number of grid points on each dimension and $N = n^2$ is the total number of points. Suppose that T_X and T_Ω are complete quadtrees with $L = \log n$ levels and, without loss of generality, L is an even integer. For a fixed level ℓ between 0 and L, the quadtree T_X has 4^ℓ nodes at level ℓ. By defining $\mathcal{I}^\ell = \{0, 1, \ldots, 4^\ell - 1\}$, we denote these nodes by A_i^ℓ with $i \in \mathcal{I}^\ell$. These 4^ℓ nodes at level ℓ are further ordered according to a Z-order curve (or Morton order) as illustrated in Fig. 1. Based on this Z-ordering, the node A_i^ℓ at level ℓ has four child nodes denoted by $A_{4i+t}^{\ell+1}$ with $t = 0, \ldots, 3$. The nodes plotted in Fig. 1 for $\ell = 1$ (middle) and $\ell = 2$ (right) illustrate the relationship between the parent node and its child nodes. Similarly, in the quadtree T_Ω, the nodes at the $L - \ell$ the are denoted as $B_j^{L-\ell}$ for $j \in \mathcal{I}^{L-\ell}$.

For any level ℓ between 0 and L, the kernel matrix K can be partitioned into $O(N)$ submatrices $K_{A_i^\ell, B_j^{L-\ell}} := (K(x, \xi))_{x \in A_i^\ell, \xi \in B_j^{L-\ell}}$ for $i \in \mathcal{I}^\ell$ and $j \in \mathcal{I}^{L-\ell}$. For simplicity, we shall denote $K_{A_i^\ell, B_j^{L-\ell}}$ as $K_{i,j}^\ell$, where the superscript ℓ denotes the level in the quadtree T_X. Because of the complementary low-rank property, every submatrix $K_{i,j}^\ell$ is numerically low-rank with the rank bounded by a uniform constant r independent of N.

The two-dimensional butterfly factorization consists of two stages. The first stage computes the factorizations

$$K_{i,j}^h \approx U_{i,j}^h S_{i,j}^h \left(V_{j,i}^h \right)^*$$

for all $i, j \in \mathcal{I}^h$ at the middle level $h = L/2$, following the form (7). These factorizations can then be assembled into three sparse matrices U^h, M^h, and V^h to give rise to a factorization for K:

$$K \approx U^h M^h \left(V^h \right)^*. \tag{10}$$

This stage is referred to as the *middle level factorization* and is described in Section 2.3. In the second stage, we recursively factorize the left and right factors U^h and V^h to obtain

$$U^h \approx U^L G^{L-1} \cdots G^h \quad \text{and} \quad \left(V^h \right)^* \approx \left(H^h \right)^* \cdots \left(H^{L-1} \right)^* \left(V^L \right)^*,$$

where the matrices on the right hand side in each formula are sparse matrices with $O(N)$ nonzero entries. Once they are ready, we assemble all factors together to produce a data-sparse approximate factorization for K:

$$K \approx U^L G^{L-1} \cdots G^h M^h \left(H^h\right)^* \cdots \left(H^{L-1}\right)^* \left(V^L\right)^*, \tag{11}$$

This stage is referred to as the *recursive factorization* and is discussed in Section 2.4.

2.3. Middle level factorization

Recall that we consider the construction of multidimensional butterfly factorization for two cases:

(i) A black-box routine for rapidly computing Kg and K^*g in $O(N \log N)$ operations is available;
(ii) A routine for evaluating any entry of K in $O(1)$ operations is given.

In Case (i), we construct an approximate rank-r SVD of each $K_{i,j}^h \in \mathbb{R}^{n \times n}$ with $i, j \in \mathcal{I}^h$ using the SVD via random matrix–vector multiplication (the first option in Section 2.1). This requires applying each $K_{i,j}^h$ to a Gaussian random matrix $C_j \in \mathbb{C}^{n \times (r+k)}$ and its adjoint to a Gaussian random matrix $R_i \in \mathbb{C}^{(r+k) \times n}$. Here r is the desired numerical rank and k is the oversampling parameter. If a black box routine for applying the matrix K and its adjoint is available, this can be done in an efficient way as follows. For each $j \in \mathcal{I}^h$, one constructs a zero-padded random matrix $C_j^P \in \mathbb{C}^{N \times (r+k)}$ by padding zero to C_j. From the relationship

$$KC_j^P = K \begin{pmatrix} 0 \\ C_j \\ 0 \end{pmatrix} = \begin{pmatrix} K_{0,j}^h C_j \\ \vdots \\ K_{4^h-1,j}^h C_j \end{pmatrix}, \tag{12}$$

it is clear that applying K to the matrix C_j^P produces $K_{i,j}^h C_j$ for all $i \in \mathcal{I}^h$. Similarly, we construct zero-padded random matrices $R_i^P \in \mathbb{C}^{N \times (r+k)}$ by padding zero to R_i and compute

$$K^* R_i^P = K^* \begin{pmatrix} 0 \\ R_i \\ 0 \end{pmatrix} = \begin{pmatrix} \left(K_{i,0}^h\right)^* R_i \\ \vdots \\ \left(K_{i,4^h-1}^h\right)^* R_i \end{pmatrix} \tag{13}$$

by using the black-box routine for applying the adjoint of K. Finally, the approximated rank-r SVD of $K_{i,j}^h$ for each pair of $i \in \mathcal{I}^h$ and $j \in \mathcal{I}^h$ is computed from $K_{i,j}^h C_j$ and $\left(K_{i,j}^h\right)^* R_i$.

In Case (ii), since an arbitrary entry of K can be evaluated in $O(1)$ operations, the approximate rank-r SVD of $K_{i,j}^h$ is computed using the SVD via randomized sampling [3,19] (the second option in Section 2.1).

In both cases, once the approximate rank-r SVD is ready, we transform it into the form of (7):

$$K_{i,j}^h \approx U_{i,j}^h S_{i,j}^h \left(V_{j,i}^h\right)^*. \tag{14}$$

Here the columns of the left and right factors $U_{i,j}^h$ and $V_{j,i}^h$ are scaled by the singular values of $K_{i,j}^h$ such that $U_{i,j}^h$ and $V_{j,i}^h$ keep track of the importance of the column and row bases for further factorizations.

After computing the rank-r factorization in (14) for all i and j in $\mathcal{I}^h$, we assemble all left factors $U_{i,j}^h$ into a matrix U^h, all middle factors into a matrix M^h, and all right factors into a matrix V^h so that

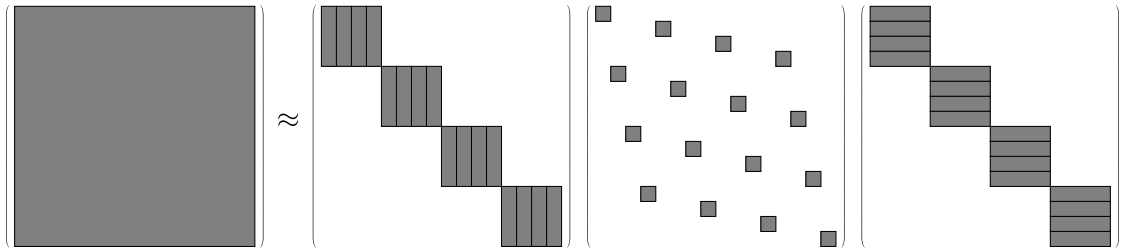$$K \approx U^h M^h (V^h)^*. \tag{15}$$

Fig. 2. The middle level factorization of a complementary low-rank matrix $K \approx U^2 M^2 (V^2)^*$ where $N = n^2 = 4^2$ and $r = 1$. Grey blocks indicate nonzero blocks. U^2 and V^2 are block-diagonal matrices with 4 blocks. The diagonal blocks of U^2 and V^2 are assembled according to Equation (16) and (17) as indicated by gray rectangles. M^2 is a 4×4 block matrix with each block $M_{i,j}^2$ itself being an 4×4 block matrix containing diagonal weight matrix on the (j, i) block.

Here U^h is a block diagonal matrix of size $N \times rN$ with n diagonal blocks U_i^h of size $n \times rn$:

$$U^h = \begin{pmatrix} U_0^h & & & \\ & U_1^h & & \\ & & \ddots & \\ & & & U_{4^h-1}^h \end{pmatrix},$$

where each diagonal block U_i^h consists of the left factors $U_{i,j}^h$ for all j as follows:

$$U_i^h = \begin{pmatrix} U_{i,0}^h & U_{i,1}^h & \cdots & U_{i,4^h-1}^h \end{pmatrix} \in \mathbb{C}^{n \times rn}. \tag{16}$$

Similarly, V^h is a block diagonal matrix of size $N \times rN$ with n diagonal blocks V_j^h of size $n \times rn$, where each diagonal block V_j^h consists of the right factors $V_{j,i}^h$ for all i as follows:

$$V_j^h = \begin{pmatrix} V_{j,0}^h & V_{j,1}^h & \cdots & V_{j,4^h-1}^h \end{pmatrix} \in \mathbb{C}^{n \times rn}. \tag{17}$$

The middle matrix $M^h \in \mathbb{C}^{rN \times rN}$ is an $n \times n$ block matrix. The (i,j)-th block $M_{i,j}^h \in \mathbb{C}^{rn \times rn}$ is itself an $n \times n$ block matrix. The only nonzero block of $M_{i,j}^h$ is the (j, i)-th block, which is equal to the $r \times r$ matrix $S_{i,j}^h$, and the other blocks of $M_{i,j}^h$ are zero. We refer to Fig. 2 for a simple example of the middle level factorization when $N = 4^2$.

2.4. Recursive factorization

In this section, we shall discuss how to recursively factorize

$$U^\ell \approx U^{\ell+1} G^\ell \tag{18}$$

and

$$(V^\ell)^* \approx (H^\ell)^* (V^{\ell+1})^* \tag{19}$$

for $\ell = h, h+1, \ldots, L-1$. After these recursive factorizations, we can construct the two-dimensional butterfly factorization

$$K \approx U^L G^{L-1} \cdots G^h M^h (H^h)^* \cdots (H^{L-1})^* (V^L)^* \tag{20}$$

by substituting these recursive factorizations into (15).

2.4.1. Recursive factorization of U^h

In the middle level factorization, we utilized the low-rank property of $K_{i,j}^h$, the kernel matrix restricted in the domain $A_i^h \times B_j^h \in T_X \times T_\Omega$, to obtain $U_{i,j}^h$ for $i, j \in \mathcal{I}^h$. We shall now use the complementary low-rank property at level $\ell = h + 1$, i.e., the matrix $K_{i,j}^{h+1}$ restricted in $A_i^{h+1} \times B_j^{h-1} \in T_X \times T_\Omega$ is numerical low-rank for $i \in \mathcal{I}^{h+1}$ and $j \in \mathcal{I}^{h-1}$. These factorizations of the column bases from level h generate the column bases at level $h + 1$ through the following four steps: splitting, merging, truncating, and assembling.

Splitting. In the middle level factorization, we have constructed

$$U^h = \begin{pmatrix} U_0^h & & & \\ & U_1^h & & \\ & & \ddots & \\ & & & U_{4^h-1}^h \end{pmatrix} \quad \text{with} \quad U_i^h = \begin{pmatrix} U_{i,0}^h & U_{i,1}^h & \cdots & U_{i,4^h-1}^h \end{pmatrix} \in \mathbb{C}^{n \times rn},$$

where each $U_{i,j}^h \in \mathbb{C}^{n \times r}$. Each node A_i^h in the quadtree T_X on the level h has four child nodes on the level $h + 1$, denoted by $\{A_{4i+t}^{h+1}\}_{t=0,1,2,3}$. According to this structure, one can split $U_{i,j}^h$ into four parts in the row space,

$$U_{i,j}^h = \begin{pmatrix} U_{i,j}^{h,0} \\ \hline U_{i,j}^{h,1} \\ \hline U_{i,j}^{h,2} \\ \hline U_{i,j}^{h,3} \end{pmatrix}, \tag{21}$$

where $U_{i,j}^{h,t}$ approximately spans the column space of the submatrix of K restricted to $A_{4i+t}^{h+1} \times B_j^h$ for each $t = 0, \ldots, 3$. Combining this with the definition of U_i^h gives rise to

$$U_i^h = \begin{pmatrix} U_{i,0}^h & U_{i,1}^h & \cdots & U_{i,4^h-1}^h \end{pmatrix} = \begin{pmatrix} U_{i,0}^{h,0} & U_{i,1}^{h,0} & \cdots & U_{i,4^h-1}^{h,0} \\ \hline U_{i,0}^{h,1} & U_{i,1}^{h,1} & \cdots & U_{i,4^h-1}^{h,1} \\ \hline U_{i,0}^{h,2} & U_{i,1}^{h,2} & \cdots & U_{i,4^h-1}^{h,2} \\ \hline U_{i,0}^{h,3} & U_{i,1}^{h,3} & \cdots & U_{i,4^h-1}^{h,3} \end{pmatrix} =: \begin{pmatrix} U_i^{h,0} \\ \hline U_i^{h,1} \\ \hline U_i^{h,2} \\ \hline U_i^{h,3} \end{pmatrix}, \tag{22}$$

where $U_i^{h,t}$ approximately spans the column space of the matrix K restricted to $A_{4i+t}^{h+1} \times \Omega$.

Merging. The merging step merges adjacent matrices $U_{i,j}^{h,t}$ in the column space to obtain low-rank matrices. For any $i \in \mathcal{I}^h$ and $j \in \mathcal{I}^{h-1}$, the merged matrix

$$\begin{pmatrix} U_{i,4j+0}^{h,t} & U_{i,4j+1}^{h,t} & U_{i,4j+2}^{h,t} & U_{i,4j+3}^{h,t} \end{pmatrix} \in \mathbb{C}^{n/4 \times 4r} \tag{23}$$

approximately spans the column space of $K_{4i+t,j}^{h+1}$ corresponding to the domain $A_{4i+t}^{h+1} \times B_j^{h-1}$. By the complementary low-rank property of the matrix K, we know $K_{4i+t,j}^{h+1}$ is numerically low-rank. Hence, the matrix in (23) is also a numerically low-rank matrix. This is the merging step equivalent to moving from level h to level $h - 1$ in T_Ω.

Truncating. The third step computes its rank-r approximation using the standard truncated SVD and putting it to the form of (8). For each $i \in \mathcal{I}^h$ and $j \in \mathcal{I}^{h-1}$, the factorization

Fig. 3. The recursive factorization of U^2 in Fig. 2. Left matrix: U^2 with each diagonal block partitioned into smaller blocks according to Equation (21) as indicated by black rectangles; Middle-left matrix: low-rank approximations of submatrices in U^2 given by Equation (24); Middle right matrix: U^3; Right matrix: G^2.

$$\begin{pmatrix} U_{i,4j+0}^{h,t} & U_{i,4j+1}^{h,t} & U_{i,4j+2}^{h,t} & U_{i,4j+3}^{h,t} \end{pmatrix} \approx U_{4i+t,j}^{h+1} G_{4i+t,j}^{h}, \tag{24}$$

defines $U_{4i+t,j}^{h+1} \in \mathbb{C}^{n/4 \times r}$ and $G_{4i+t,j}^{h} \in \mathbb{C}^{r \times 4r}$.

Assembling In the final step, we construct the factorization $U^h \approx U^{h+1}G^h$ using (24). Since $\mathcal{I}^{h+1}$ is the same as $\{4i + t\}_{i \in \mathcal{I}^h, t=0,1,2,3}$, one can arrange (24) for all i and j into a single formula as follows:

$$U^h \approx U^{h+1}G^h =$$

$$\begin{pmatrix} U_0^{h+1} & & & & & & & \\ & \ddots & & & & & & \\ & & U_3^{h+1} & & & & & \\ & & & U_4^{h+1} & & & & \\ & & & & \ddots & & & \\ & & & & & U_7^{h+1} & & \\ & & & & & & \ddots & \\ & & & & & & & U_{4^{h+1}-4}^{h+1} \\ & & & & & & & & \ddots \\ & & & & & & & & & U_{4^{h+1}-1}^{h+1} \end{pmatrix} \begin{pmatrix} G_0^h \\ \vdots \\ G_3^h \\ & G_4^h \\ & \vdots \\ & G_7^h \\ & & \ddots \\ & & & G_{4^{h+1}-4}^h \\ & & & \vdots \\ & & & G_{4^{h+1}-1}^h \end{pmatrix},$$

where the blocks are given by

$$U_i^{h+1} = \begin{pmatrix} U_{i,0}^{h+1} & U_{i,1}^{h+1} & \cdots & U_{i,4^{h-1}-1}^{h+1} \end{pmatrix}$$

and

$$G_i^h = \begin{pmatrix} G_{i,0}^h & & & \\ & G_{i,1}^h & & \\ & & \ddots & \\ & & & G_{i,4^{h-1}-1}^h \end{pmatrix}$$

for $i \in \mathcal{I}^{h+1}$. Fig. 3 shows a toy example of the recursive factorization of U^h when $N = 4^2$, $h = 2$ and $r = 1$. Since there are $O(1)$ nonzero entries in each $G_{i,j}^h$ and $O(4^{h+1} \cdot 4^{h-1}) = O(N)$ such matrices, there are only $O(N)$ nonzero entries in G^h.

In a similar way, we can now factorize $U^\ell \approx U^{\ell+1}G^\ell$ for $h < \ell \leq L - 1$. As before, the key point is that the columns of

$$\begin{pmatrix} U_{i,4j+0}^{\ell,t} & U_{i,4j+1}^{\ell,t} & U_{i,4j+2}^{\ell,t} & U_{i,4j+3}^{\ell,t} \end{pmatrix} \tag{25}$$

approximately span the column space of $K_{4i+t,j}^{\ell+1}$, which is of rank r numerically due to the complementary low-rank property. Computing its rank-r approximation via the standard truncated SVD results in a form of (8)

$$\begin{pmatrix} U_{i,4j+0}^{\ell,t} & U_{i,4j+1}^{\ell,t} & U_{i,4j+2}^{\ell,t} & U_{i,4j+3}^{\ell,t} \end{pmatrix} \approx U_{4i+t,j}^{\ell+1} G_{4i+t,j}^{\ell} \tag{26}$$

for $i \in \mathcal{I}^{\ell}$ and $j \in \mathcal{I}^{L-\ell-1}$. After assembling these factorizations together, we obtain

$$U^{\ell} \approx U^{\ell+1} G^{\ell} =$$

$$\begin{pmatrix} U_0^{\ell+1} & & & & & & \\ & \ddots & & & & & \\ & & U_3^{\ell+1} & & & & \\ & & & U_4^{\ell+1} & & & \\ & & & & \ddots & & \\ & & & & & U_7^{\ell+1} & & \\ & & & & & & \ddots & \\ & & & & & & & U_{4^{\ell+1}-4}^{\ell+1} & \\ & & & & & & & & \ddots \\ & & & & & & & & & U_{4^{\ell+1}-1}^{\ell+1} \end{pmatrix} \begin{pmatrix} G_0^{\ell} & & & & & \\ \vdots & & & & & \\ G_3^{\ell} & & & & & \\ & G_4^{\ell} & & & & \\ & \vdots & & & & \\ & G_7^{\ell} & & & & \\ & & \ddots & & & \\ & & & G_{4^{\ell+1}-4}^{\ell} & \\ & & & \vdots & \\ & & & G_{4^{\ell+1}-1}^{\ell} \end{pmatrix},$$

where

$$U_i^{\ell+1} = \begin{pmatrix} U_{i,0}^{\ell+1} & U_{i,1}^{\ell+1} & \cdots & U_{i,4^{L-\ell-1}-1}^{\ell+1} \end{pmatrix}$$

and

$$G_i^{\ell} = \begin{pmatrix} G_{i,0}^{\ell} & & & \\ & G_{i,1}^{\ell} & & \\ & & \ddots & \\ & & & G_{i,4^{L-\ell-1}-1}^{\ell} \end{pmatrix}$$

for $i \in \mathcal{I}^{\ell+1}$.

After the $L - h$ step of recursive factorizations $U^{\ell} \approx U^{\ell+1} G^{\ell}$ for $\ell = h, h+1, \ldots, L-1$, the recursive factorization of U^h takes the following form:

$$U^h \approx U^L G^{L-1} \cdots G^h. \tag{27}$$

Similarly to the analysis of G^h, it is also easy to check that there are only $O(N)$ nonzero entries in each G^{ℓ} in (27). As to the first factor U^L, it has $O(N)$ nonzero entries since there are $O(N)$ diagonal blocks in U^L and each block contains $O(1)$ entries.

2.4.2. Recursive factorization of V^h

The recursive factorization of V^{ℓ} is similar to that of U^{ℓ} for $\ell = h, h+1, \ldots, L-1$. At each level ℓ, we benefit from the fact that

$$\begin{pmatrix} V_{j,4i+0}^{\ell,t} & V_{j,4i+1}^{\ell,t} & V_{j,4i+2}^{\ell,t} & V_{j,4i+3}^{\ell,t} \end{pmatrix}$$

approximately spans the row space of $K_{i,4j+t}^{L-\ell-1}$ and hence is numerically low-rank for $j \in \mathcal{I}^{L-\ell}$ and $i \in \mathcal{I}^{\ell-1}$. Applying the same procedure in Section 2.4.1 to V^h leads to

$$V^h \approx V^L H^{L-1} \cdots H^h. \tag{28}$$

2.5. Complexity analysis

By combining the results of the middle level factorization in (15) and the recursive factorizations in (27) and (28), we obtain the final butterfly factorization

$$K \approx U^L G^{L-1} \cdots G^h M^h \left(H^h\right)^* \cdots \left(H^{L-1}\right)^* \left(V^L\right)^*, \tag{29}$$

each factor of which contains $O(N)$ nonzero entries. We refer to Fig. 4 for an illustration of the butterfly factorization of K when $N = 16^2$.

The complexity of constructing the butterfly factorization comes from two parts: the middle level factorization and the recursive factorization. For the middle level factorization, the construction cost is different depending on which of the two cases mentioned in Section 2.3 is under consideration, since they use different approaches in constructing rank-r SVDs at the middle level.

- In Case (i), the dominant cost is to apply K and K^* to $N^{1/2}$ Gaussian random matrices of size $N \times O(1)$. Assuming that the given black-box routine for applying K and K^* to a vector takes $O(C_K(N))$ operations, the total operation complexity is $O(C_K(N)N^{1/2})$.
- In Case (ii), we apply the SVD procedure with random sampling to N submatrices of size $N^{1/2} \times N^{1/2}$. Since the operation complexity for each submatrix is $O(N^{1/2})$, the overall complexity is $O(N^{3/2})$.

In the recursive factorization stage, most of the work comes from factorizing U^h and V^h. There are $O(\log N)$ stages appeared in the factorization of U^h. At the ℓ stage, the matrix U^ℓ to be factorized consists of 4^ℓ diagonal blocks. There are $O(N)$ factorizations and each factorization takes $O(N/4^\ell)$ operations. Hence, the operation complexity to factorize U^ℓ is $O(N^2/4^\ell)$. Summing up all the operations in each step yields the overall operation complexity for recursively factorizing U^h:

$$\sum_{\ell=h}^{L-1} O(N^2/4^\ell) = O(N^{3/2}). \tag{30}$$

The peak of the memory usage of the butterfly factorization is due to the middle level factorization where we need to store the results of $O(N)$ factorizations of size $O(N^{1/2})$. Hence, the memory complexity for the two-dimensional butterfly factorization is $O(N^{3/2})$. For Case (ii), one can actually do better by following the same argument in [7]. One can interleave the order of generation and recursive factorization of $U_{i,j}^h$ and $V_{j,i}^h$. By factorizing $U_{i,j}^h$ and $V_{j,i}^h$ individually instead of formulating (15), the memory complexity in Case (ii) can be reduced to $O(N \log N)$.

The cost of applying the butterfly factorization is equal to the number of nonzero entries in the final factorization, which is $O(N \log N)$. Table 1 summarizes the complexity analysis for the two-dimensional butterfly factorization.

2.6. Extensions

We have introduced the two-dimensional butterfly factorization for a complementary low-rank kernel matrix K in the entire domain $X \times \Omega$. Although we have assumed the uniform grid in (4) and (5), the butterfly factorization extends naturally to more general settings.
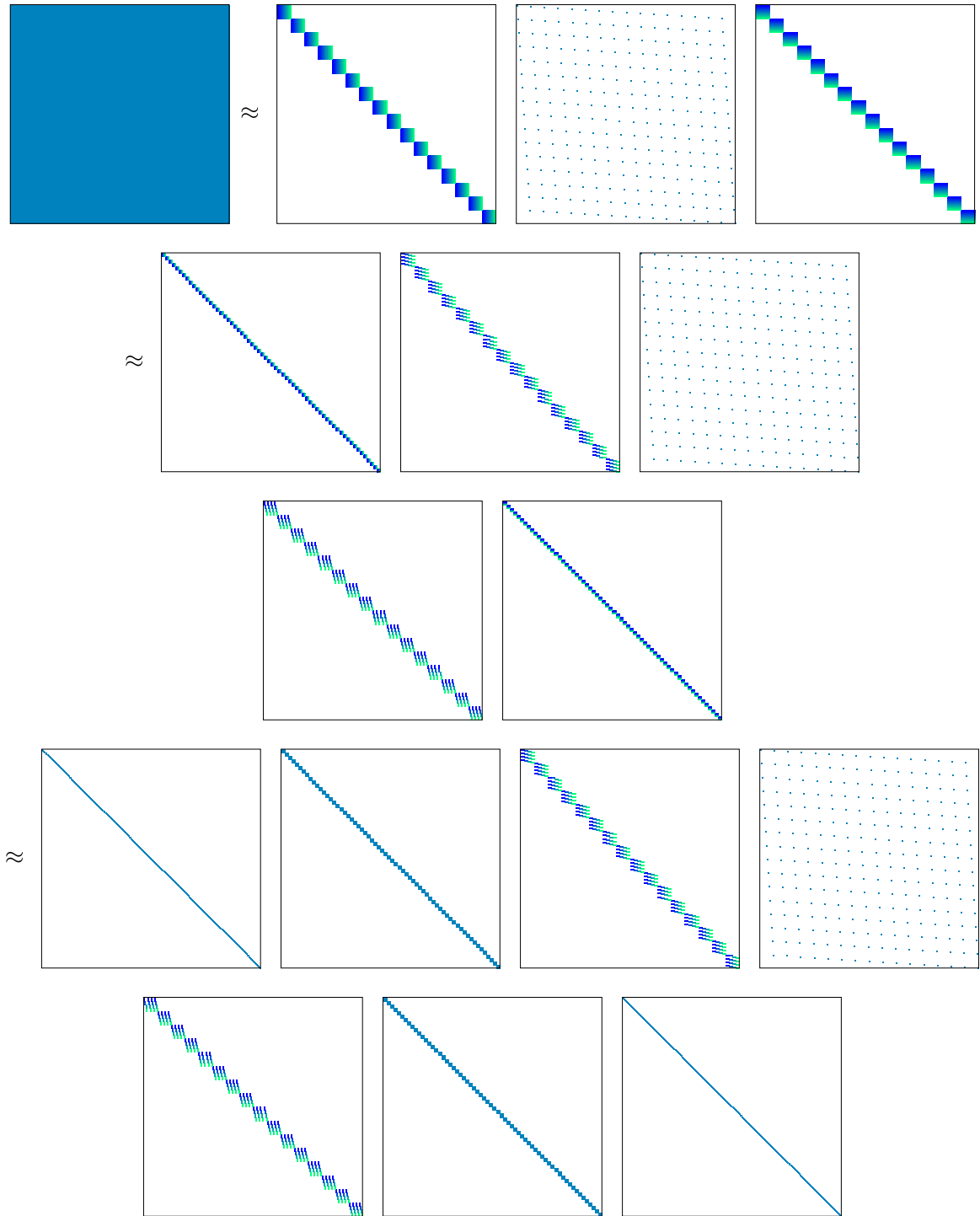
Fig. 4. A full butterfly factorization for a two dimensional problem of size 16^2 and fixed rank $r = 1$. The above figure visualizes the matrices in $K \approx U^3 M^3 (V^3)^* \approx U^4 G^3 M^3 (H^3)^* (V^4)^* \approx U^5 G^4 G^3 M^3 (H^3)^* (H^4)^* (V^5)^*$.

In the case with non-uniform point sets X or Ω, one can still construct a butterfly factorization for K following the same procedure. More specifically, we still construct two trees T_X and T_Ω *adaptively* via hierarchically partitioning the square domains covering X and Ω. For non-uniform point sets X and Ω, the

Table 1
The time and memory complexity of the two-dimensional butterfly factorization. Here $C_K(N)$ is the complexity of applying the matrices K and K^* to a vector. For most butterfly algorithms, $C_K(N) = O(N \log N)$.

		SVD via rand. matvec	SVD via rand. sampling
Factorization complexity	Middle level factorization	$O(C_K(N)N^{1/2})$	$O(N^{3/2})$
	Recursive factorization	$O(N^{3/2})$	
	Total	$O(C_K(N)N^{1/2})$	$O(N^{3/2})$
Memory complexity		$O(N^{3/2})$	$O(N \log N)$
Application complexity		$O(N \log N)$	

numbers of points in A_i^ℓ and $B_j^{L-\ell}$ are different. If a node does not contain any point inside it, it is simply discarded from the quadtree.

The complexity analysis summarized in Table 1 remains valid in the case of non-uniform point sets X and Ω. On each level $\ell = h, \ldots, L$ of the butterfly factorization, although the sizes of low-rank submatrices are different, the total number of submatrices and the numerical rank remain the same. Hence, the total operation and memory complexity remains the same as summarized in Table 1.

3. Polar butterfly factorization

In Section 2, we have introduced a two-dimensional butterfly factorization for a complementary low-rank kernel matrix K in the entire domain $X \times \Omega$. In this section, we will introduce a polar butterfly factorization to deal with the kernel function $K(x, \xi) = e^{2\pi \iota \Phi(x,\xi)}$. Such a kernel matrix has a singularity at $\xi = 0$ and the approach taken here follows the polar butterfly algorithm proposed in [2].

3.1. Polar butterfly algorithm

The multidimensional Fourier integral operator (FIO) is defined as

$$u(x) = \sum_{\xi \in \Omega} e^{2\pi \iota \Phi(x,\xi)} g(\xi), \quad x \in X, \tag{31}$$

where the phase function $\Phi(x, \xi)$ is assumed to be real-analytic in (x, ξ) for $\xi \neq 0$, and is homogeneous of degree 1 in ξ, namely, $\Phi(x, \lambda\xi) = \lambda\Phi(x, \xi)$ for all $\lambda > 0$. Here the grids X and Ω are the same as those in (4) and (5).

As the phase function $\Phi(x, \xi)$ is singular at $\xi = 0$, the numerical rank of the kernel $e^{2\pi \iota \Phi(x,\xi)}$ in a domain near or containing $\xi = 0$ is typically large. Hence, in general $K(x, \xi) = e^{2\pi \iota \Phi(x,\xi)}$ does not satisfy the complementary low-rank property over the domain $X \times \Omega$ with quadtree structures T_X and T_Ω. To fix this problem, the polar butterfly algorithm introduces a scaled polar transformation on Ω:

$$\xi = (\xi_1, \xi_2) = \frac{\sqrt{2}}{2} n p_1 \cdot (\cos 2\pi p_2, \sin 2\pi p_2), \tag{32}$$

for $\xi \in \Omega$ and $p = (p_1, p_2) \in [0, 1]^2$. In the rest of this section, we use p to denote a point in the polar coordinate and P for the set of all points p transformed from $\xi \in \Omega$. This transformation gives rise to a new phase function $\Psi(x, p)$ in variables x and p satisfying

$$\Psi(x, p) = \frac{1}{n}\Phi(x, \xi(p)) = \frac{\sqrt{2}}{2}\Phi(x, (\cos 2\pi p_2, \sin 2\pi p_2)) \cdot p_1, \tag{33}$$

where the last equality comes from the fact that $\Phi(x, \xi)$ is homogeneous of degree 1 in ξ. This new phase function $\Psi(x, p)$ is smooth in the entire domain $X \times P$ and the FIO in (31) takes the new form

$$u(x) = \sum_{p \in P} e^{2\pi i n \Psi(x,p)} g(p), \quad x \in X. \tag{34}$$

The transformation (32) ensures that $X \times P \subset [0,1]^2 \times [0,1]^2$. By partitioning $[0,1]^2$ recursively, we can construct two quadtrees T_X and T_P of depth $L = O(\log n)$ for X and P, respectively. The following theorem is a rephrased version of Theorem 3.1 in [2] that shows analytically the complementary low-rank property of $e^{2\pi i n \Psi(x,p)}$ in the (X,P) domain.

Theorem 3.1. *Suppose A is a node in T_X at level ℓ and B is a node in T_P at level $L - \ell$. Given an FIO kernel function $e^{2\pi i n \Psi(x,p)}$ with a real-analytic phase function in the joint variables x and p, there exist $\epsilon_0 > 0$ and $n_0 > 0$ such that for any positive $\epsilon \le \epsilon_0$ and $n \ge n_0$, there exist r_ϵ pairs of functions $\{\alpha_t^{A,B}(x), \beta_t^{A,B}(p)\}_{1 \le t \le r_\epsilon}$ satisfying that*

$$\left| e^{2\pi i n \Psi(x,p)} - \sum_{t=1}^{r_\epsilon} \alpha_t^{A,B}(x) \beta_t^{A,B}(p) \right| \le \epsilon,$$

for $x \in A$ and $p \in B$ with $r_\epsilon \lesssim \log^4(1/\epsilon)$.

Based on Theorem 3.1, the polar butterfly algorithm traverses upward in T_Ω and downward in T_X simultaneously and visits the low-rank submatrices $K_{A,B} = \{K(x_i, \xi_j)\}_{x_i \in A, \xi_j \in B}$ for pairs (A,B) in $T_X \times T_P$. The polar butterfly algorithm is asymptotically very efficient: for a given input vector $g(p)$ for $p \in P$, it evaluates (34) in $O(N \log N)$ steps using $O(N)$ memory space. We refer the readers to [2] for a detailed description of this algorithm.

3.2. Factorization algorithm

Combining the polar butterfly algorithm with the butterfly factorization outlined in Section 2 gives rise to the following polar butterfly factorization.

1. *Preliminary.* Take the polar transformation of each point in Ω and reformulate the problem

$$u(x) = \sum_{\xi \in \Omega} e^{2\pi i \Phi(x,\xi)} g(\xi), \quad x \in X, \tag{35}$$

   into

$$u(x) = \sum_{p \in P} e^{2\pi i n \Psi(x,p)} g(p), \quad x \in X. \tag{36}$$

2. *Factorization.* Apply the two-dimensional butterfly factorization to the kernel $e^{2\pi i n \Psi(x,p)}$ defined on a non-uniform point set in $X \times P$. The corresponding kernel matrix is approximated as

$$K \approx U^L G^{L-1} \cdots G^h M^h \left(H^h \right)^* \cdots \left(H^{L-1} \right)^* \left(V^L \right)^*. \tag{37}$$

Since the polar butterfly factorization essentially applies the original butterfly factorization to non-uniform point sets X and P, it has the same complexity as summarized in Table 1. Depending on the SVD procedure employed in the middle level factorization, we refer to it either as PBF-m (when SVD via random matrix–vector multiplication is used) or as PBF-s (when SVD via random sampling is used).

3.3. Numerical results

This section presents two numerical examples to demonstrate the efficiency of the polar butterfly factorization. The numerical results were obtained in MATLAB on a server with 2.40 GHz CPU and 1.5 TB of memory.

In this section, we denote by $\{u^p(x)\}_{x \in X}$ the results obtained via the PBF. The relative error of the PBF is estimated as follows, by comparing $u^p(x)$ with the exact values $u(x)$:

$$e^p = \sqrt{\frac{\sum_{x \in S} |u^p(x) - u(x)|^2}{\sum_{x \in S} |u(x)|^2}}, \tag{38}$$

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

Example 1. The first example is a two-dimensional generalized Radon transform that is an FIO defined as follows:

$$u(x) = \sum_{\xi \in \Omega} e^{2\pi \imath \Phi(x,\xi)} g(\xi), \quad x \in X, \tag{39}$$

with the phase function given by

$$\begin{aligned}
\Phi(x,\xi) &= x \cdot \xi + \sqrt{c_1^2(x)\xi_1^2 + c_2^2(x)\xi_2^2}, \\
c_1(x) &= (2 + \sin(2\pi x_1)\sin(2\pi x_2))/16, \\
c_2(x) &= (2 + \cos(2\pi x_1)\cos(2\pi x_2))/16,
\end{aligned} \tag{40}$$

where X and Ω are defined in (4) and (5). The computation in (39) approximately integrates over spatially varying ellipses, for which $c_1(x)$ and $c_2(x)$ are the axis lengths of the ellipse centered at the point $x \in X$. The corresponding matrix form of (39) is simply

$$u = Kg, \quad K = (e^{2\pi \imath \Phi(x,\xi)})_{x \in X, \xi \in \Omega}. \tag{41}$$

As $e^{2\pi \imath \Phi(x,\xi)}$ is known explicitly, we are able to use the PBF-s (i.e., the one with random sampling in the middle level factorization) to approximate the kernel matrix K given by $e^{2\pi \imath \Phi(x,\xi)}$. After the construction of the butterfly factorization, the summation in (39) can be evaluated efficiently by applying these sparse factors to $g(\xi)$. Table 2 summarizes the results of this example.

Example 2. The second example evaluates the composition of two FIOs with the same phase function $\Phi(x,\xi)$. This is given explicitly by

$$u(x) = \sum_{\eta \in \Omega} e^{2\pi \imath \Phi(x,\eta)} \sum_{y \in X} e^{-2\pi \imath y \cdot \eta} \sum_{\xi \in \Omega} e^{2\pi \imath \Phi(y,\xi)} g(\xi), \quad x \in X, \tag{42}$$

where the phase function is given in (40). The corresponding matrix representation is

$$u = KFKg, \tag{43}$$

where K is the matrix given in (41) and F is the matrix representation of the discrete Fourier transform. Under relatively mild assumptions (see [5] for details), the composition of two FIOs is again an FIO. Hence, the kernel matrix

Table 2
Numerical results provided by the PBF with randomized sampling algorithm for the FIO in (39). n is the number of grid points in each dimension; $N = n^2$ is the size of the kernel matrix; r is the max rank used in the low-rank approximation; $T_{f,p}$ is the factorization time of the PBF; T_p is the application time of the PBF. The last column shows the speedup factor compared to the direct evaluation.

n, r	ϵ^p	$T_{f,p}$ (min)	T_p (s)	Speedup
64, 6	2.46e−02	6.51e−01	2.37e−02	1.54e+02
128, 6	7.55e−03	9.84e+00	2.30e−01	1.67e+02
256, 6	5.10e−02	2.73e+01	6.23e−01	7.55e+02
512, 6	1.46e−02	4.00e+02	7.88e+00	4.15e+02
64, 14	7.93e−04	7.34e−01	5.98e−02	8.72e+01
128, 14	7.28e−04	1.17e+01	7.15e−01	4.28e+01
256, 14	2.15e−03	3.93e+01	1.46e+00	2.86e+02
512, 14	1.25e−03	5.63e+02	1.05e+01	3.35e+02
64, 22	6.96e−05	7.40e−01	8.24e−02	4.51e+01
128, 22	7.23e−05	1.16e+01	1.04e+00	3.69e+01
256, 22	2.44e−04	5.14e+01	5.94e+00	7.74e+01

Table 3
Numerical results provided by the PBF with randomized SVD algorithm for the composition of FIOs given in (43).

n, r	ϵ^p	$T_{f,p}$ (min)	T_p (s)	Speedup
64, 12	3.84e−02	6.22e+00	2.18e−02	3.34e+02
128, 12	1.31e−02	3.86e+02	1.80e−01	4.25e+02
64, 20	2.24e−03	8.58e+00	3.04e−02	2.39e+02
128, 20	2.23e−03	3.68e+02	3.60e−01	2.13e+02

$$\widetilde{K} := KFK \tag{44}$$

of the product can be approximated by the butterfly factorization. Notice that the kernel function of $\widetilde{K}$ defined by (44) is not given explicitly. However, (44) provides fast algorithms for applying $\widetilde{K}$ and its adjoint through the fast algorithms for K and F. For example, the butterfly factorization of Example 1 enables the efficient application of K and K^* in $O(N \log N)$ operations. Applying of F and F^* can be done by the fast Fourier transform in $O(N \log N)$ operations. Therefore, we can apply the PBF-m (i.e., the one with random matrix–vector multiplication) to factorize the kernel $\widetilde{K} = KFK$. Table 3 summarizes the numerical results of this example, the composing of two FIOs.

Discussion. The numerical results in Tables 2 and 3 support the asymptotic complexity analysis. When we fix r and let n grow, the actually running time fluctuates around the asymptotic scaling since the implementation of the algorithms differ slightly depending on whether L is odd or even. However, the overall trend matches well with the $O(N^{3/2})$ construction cost and the $O(N \log N)$ application cost. For a fixed n, one can improve the accuracy by increasing the truncation rank r. From the tables, one observes that the relative error decreases by a factor of 10 when we increase the rank r by 8 every time. In the second example, since the composition of two FIOs typically has higher ranks compared to a single FIO, the numerical rank r used for the composition is larger than that for a single FIO in order to maintain comparable accuracy.

4. Multiscale butterfly factorization

In this section, we discuss yet another approach for constructing butterfly factorization for the kernel $K(x, \xi) = e^{2\pi \imath \Phi(x, \xi)}$ with singularity at $\xi = 0$. This is based on the multiscale butterfly algorithm introduced in [8].
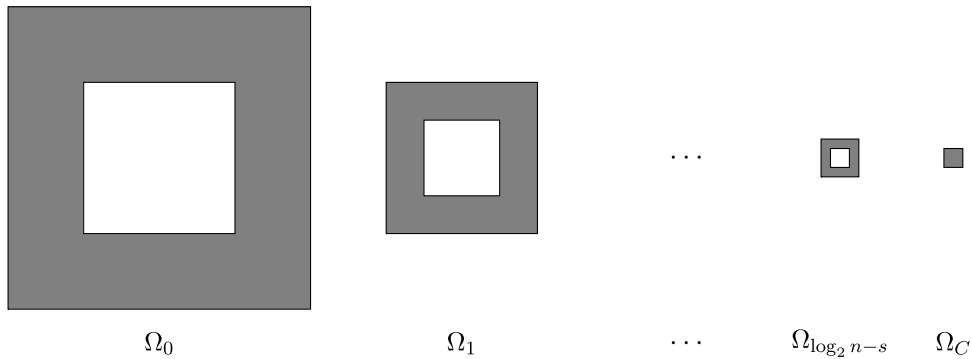
$$\Omega_0 \qquad\qquad \Omega_1 \qquad \cdots \qquad \Omega_{\log_2 n - s} \quad \Omega_C$$

Fig. 5. This figure shows the frequency domain decomposition of Ω. Each subdomain Ω_t, $t = 0, 1, \ldots, \log_2 n - s$, is a corona subdomain and Ω_C is a small square subdomain covering the origin.

4.1. Multiscale butterfly algorithm

The key idea of the multiscale butterfly algorithm [8] is to hierarchically partition the domain Ω into subdomains excluding the singular point $\xi = 0$. This multiscale partition is illustrated in Fig. 5 with

$$\Omega_t = \left\{ (\xi_1, \xi_2) : \frac{n}{2^{t+2}} < \max(|\xi_1|, |\xi_2|) \leq \frac{n}{2^{t+1}} \right\} \cap \Omega, \tag{45}$$

for $t = 0, 1, \ldots, \log_2 n - s$, s is a small constant, and $\Omega_C = \Omega \backslash \cup_t \Omega_t$. Equation (45) is a corona decomposition of Ω, where each Ω_t is a corona subdomain and Ω_C is a square subdomain at the center containing $O(1)$ points.

The FIO kernel $e^{2\pi\imath\Phi(x,\xi)}$ satisfies the complementary low-rank property when it is restricted in each subdomain $X \times \Omega_t$. This observation is supported by the following theorem rephrased from Theorem 3.1 in [8]. Here the notation $\mathrm{dist}(B, 0) = \min_{\xi \in B} \|\xi - 0\|$ is the distance between the square B and the origin $\xi = 0$ in Ω.

Theorem 4.1. *Given an FIO kernel function $e^{2\pi\imath\Phi(x,\xi)}$ with a real-analytic phase function $\Phi(x,\xi)$ for x and ξ away from $\xi = 0$, there exist a constant $n_0 > 0$ and a small constant ϵ_0 such that the following statement holds. Let A and B be two squares in X and Ω with sidelength w_A and w_B, respectively. Suppose $w_A w_B \leq 1$ and $\mathrm{dist}(B, 0) \geq \frac{n}{4}$. For any positive $\epsilon \leq \epsilon_0$ and $n \geq n_0$, there exist r_ϵ pairs of functions $\{\alpha_t^{A,B}(x), \beta_t^{A,B}(p)\}_{1 \leq t \leq r_\epsilon}$ satisfying that*

$$\left| e^{2\pi\imath\Phi(x,\xi)} - \sum_{t=1}^{r_\epsilon} \alpha_t^{A,B}(x)\beta_t^{A,B}(\xi) \right| \leq \epsilon,$$

for $x \in A$ and $\xi \in B$ with $r_\epsilon \lesssim \log^4(1/\epsilon)$.

According to the low-rank property in Theorem 4.1, the multiscale butterfly algorithm rewrites (31) as a multiscale summation,

$$u(x) = u_C(x) + \sum_{t=0}^{\log_2 n - s} u_t(x) = \sum_{\xi \in \Omega_C} e^{2\pi\imath\Phi(x,\xi)} g(\xi) + \sum_{t=0}^{\log_2 n - s} \sum_{\xi \in \Omega_t} e^{2\pi\imath\Phi(x,\xi)} g(\xi). \tag{46}$$

For each t, the multiscale butterfly factorization algorithm evaluates $u_t(x) = \sum_{\xi \in \Omega_t} e^{2\pi\imath\Phi(x,\xi)} g(\xi)$ with a standard butterfly algorithm such as the one that relies on the oscillatory Lagrange interpolation on Chebyshev grid (see [2]). The final piece $u_C(x)$ is evaluated directly in $O(N)$ operations. As a result, the

multiscale butterfly algorithm asymptotically takes $O(N \log N)$ operations to evaluate (46) for a given input function $g(\xi)$ for $\xi \in \Omega$. We refer the reader to [8] for the detailed exposition.

4.2. Factorization algorithm

Combining the multiscale butterfly algorithm with the butterfly factorization outlined in Section 2 gives rise to the following multiscale butterfly factorization:

1. *Preliminary.* Decompose domain Ω into subdomains as in (45). Reformulate the problem into a multi-scale summation according to (46):

$$K = K_C R_C + \sum_{t=0}^{\log_2 n - s} K_t R_t. \tag{47}$$

   Here K_C and K_t are kernel matrices corresponding to $X \times \Omega_C$ and $X \times \Omega_t$. R_C and R_t are the restriction operators to the domains Ω_C and Ω_t respectively.

2. *Factorization.* Recall that $L = \log_2 n$. For each $t = 0, 1, \ldots, L - s$, apply the two-dimensional butterfly factorization on $K(x, \xi) = e^{2\pi i \Phi(x,\xi)}$ restricted in $X \times \Omega_t$. Let $\widetilde{\Omega}_t$ be the smallest square that contains Ω_t. Define $L_t = 2\lfloor (L - t)/2 \rfloor$, where $\lfloor \cdot \rfloor$ is the largest integer less than or equal to a given number. We construct two quadtrees T_X and $T_{\widetilde{\Omega}_t}$ of depth L_j with X and $\widetilde{\Omega}_t$ being the roots, respectively. Applying the two-dimensional butterfly factorization using the quadtrees T_X and $T_{\widetilde{\Omega}_t}$ gives the t-th butterfly factorization:

$$K_t \approx U_t^{L_t} G_t^{L_t-1} \cdots G_t^{\frac{L_t}{2}} M_t^{\frac{L_t}{2}} \left(H_t^{\frac{L_t}{2}} \right)^* \cdots \left(H_t^{L_t-1} \right)^* \left(V_t^{L_t} \right)^*.$$

   Note that $1/4$ of the tree $T_{\widetilde{\Omega}_t}$ is empty and we can simply ignore the computation for these parts. This is a special case of non-uniform point sets. Once we have computed all butterfly factorizations, the multiscale summation in (47) is approximated by

$$K \approx K_C R_C + \sum_{t=0}^{L-s} U_t^{L_t} G_t^{L_t-1} \cdots M_t^{\frac{L_t}{2}} \cdots \left(H_t^{L_t-1} \right)^* \left(V_t^{L_t} \right)^* R_t. \tag{48}$$

The idea of the hierarchical decomposition of Ω not only avoids the singularity of $K(x, \xi)$ at $\xi = 0$, but also maintains the efficiency of the butterfly factorization. The butterfly factorization for the kernel matrix restricted in $X \times \Omega_t$ is a special case of non-uniform butterfly factorization in which the center of Ω_t contains no point. Since the number of points in Ω_t is decreasing exponentially in t, the operation and memory complexity of the multiscale butterfly factorization is dominated by the butterfly factorization of K_t for $t = 0$, which is bounded by the complexity summarized in Table 1. Depending on the SVD procedure in the middle level factorization, we refer this factorization either as MBF-m (when SVD via random matrix–vector multiplication is used) or as MBF-s (when SVD via random sampling is used).

4.3. Numerical results

This section presents two numerical examples to demonstrate the efficiency of the MBF as well. The numerical results are obtained in the same environment as the one used in Section 3.3. Here we denote by $\{u^m(x), x \in X\}$ the results obtained via the MBF. The relative error is estimated by

Table 4
Numerical results provided by the MBF with the randomized sampling algorithm for the FIO given in (50). n is the number of grid points in each dimension; $N = n^2$ is the size of the kernel matrix; r is the max rank used in low-rank approximation; $T_{f,m}$ is the factorization time of the MBF; T_m is the application time of the MBF. The last column shows the speedup factor compared to the direct evaluation.

n, r	ϵ^m	$T_{f,m}$ (min)	T_m (s)	Speedup
64, 12	1.58e−02	4.48e−01	4.09e−02	1.13e+02
128, 12	1.47e−02	5.64e+00	1.93e−01	2.02e+02
256, 12	2.13e−02	2.16e+01	5.51e−01	9.26e+02
512, 12	1.97e−02	2.97e+02	5.07e+00	6.45e+02
64, 20	5.51e−03	4.74e−01	6.11e−02	6.17e+01
128, 20	4.27e−03	5.95e+00	5.01e−01	7.63e+01
256, 20	1.68e−03	3.03e+01	2.51e+00	1.79e+02
512, 20	2.02e−03	4.57e+02	1.14e+01	2.98e+02
64, 28	7.42e−05	7.18e−01	3.92e−02	6.23e+01
128, 28	8.46e−05	1.23e+01	5.42e−01	7.43e+01
256, 28	5.63e−04	6.73e+01	3.23e+00	1.43e+02
512, 28	4.18e−04	7.20e+02	1.66e+01	2.14e+02

$$e^m = \sqrt{\frac{\sum_{x \in S} |u^m(x) - u(x)|^2}{\sum_{x \in S} |u(x)|^2}}, \tag{49}$$

where S is a set of 256 randomly sampled from X. In the multiscale decomposition of Ω, we recursively divide Ω until the center part is of size 16 by 16.

Example 1. We revisit the first example in Section 3.3 to illustrate the performance of the MBF,

$$u(x) = \sum_{\xi \in \Omega} e^{2\pi \iota \Phi(x, \xi)} g(\xi), \quad x \in X, \tag{50}$$

with a kernel $\Phi(x, \xi)$ given by

$$\begin{aligned}
\Phi(x, \xi) &= x \cdot \xi + \sqrt{c_1^2(x)\xi_1^2 + c_2^2(x)\xi_2^2}, \\
c_1(x) &= (2 + \sin(2\pi x_1)\sin(2\pi x_2))/16, \\
c_2(x) &= (2 + \cos(2\pi x_1)\cos(2\pi x_2))/16,
\end{aligned} \tag{51}$$

where X and Ω are defined in (4) and (5). Table 4 summarizes the results of this example obtained by applying the MBF-s.

Example 2. Here we revisit the second example in Section 3.3 to illustrate the performance of the MBF. Recall that the matrix representation of a composition of two FIOs is

$$u = \widetilde{K}g = KFKg, \tag{52}$$

and that there are fast algorithms to apply K, F and their adjoints. Hence, we can apply the MBF-m (i.e., with the random matrix–vector multiplication) to factorize $\widetilde{K}$ into the form of (48). Table 5 summarizes the results.

Discussion. The results in Tables 4 and 5 agree with the $O(N^{3/2} \log N)$ complexity analysis of the construction algorithm. As we double the problem size n, the factorization time increases by a factor 9 on average.

Table 5
MBF numerical results for the composition of FIOs given in (52).

n, r	ϵ^m	$T_{f,m}$ (min)	T_m (s)	Speedup
64, 16	1.86e−02	4.05e+00	1.95e−02	4.23e+02
128, 16	1.76e−02	1.27e+02	1.86e−01	4.17e+02
64, 24	4.43e−03	5.37e+00	2.52e−02	3.27e+02
128, 24	3.02e−03	1.79e+02	2.29e−01	3.40e+02

The actual application time in these numerical examples matches the theoretical operation complexity of $O(N \log N)$. In Table 4, the relative error decreases by a factor of 10 when the increment of the rank r is 6. In Table 5, the relative error decreases by a factor of 6 when the increment of the rank r is 8.

5. Conclusion

We have introduced three multidimensional butterfly factorizations as data-sparse representations of a class of kernel matrices coming from multidimensional integral transforms. When the integral kernel $K(x, \xi)$ satisfies the complementary low-rank property in the entire domain, the butterfly factorization introduced in Section 2 represents an $N \times N$ kernel matrix as a product of $O(\log N)$ sparse matrices. In the FIO case for which the kernel $K(x, \xi)$ is singular at $\xi = 0$, we propose two extensions: (1) the polar butterfly factorization that incorporates a polar coordinate transformation to remove the singularity and (2) the multiscale butterfly factorization that relies on a hierarchical partitioning in the Ω domain. For both extensions, the resulting butterfly factorization takes $O(N \log N)$ storage space and $O(N \log N)$ steps for computing matrix–vector multiplication as before.

The butterfly factorization for higher dimensions ($d > 2$) can be constructed in a similar way. For the polar butterfly factorization, one simply applies a d-dimensional spherical transformation to the frequency domain Ω. For the multiscale butterfly factorization, one can again decompose the frequency domain as a union of dyadic shells centered round the singularity at $\xi = 0$.

Acknowledgments

References

[1] E. Candès, L. Demanet, L. Ying, Fast computation of Fourier integral operators, SIAM J. Sci. Comput. 29 (6) (2007) 2464–2493.
[2] E. Candès, L. Demanet, L. Ying, A fast butterfly algorithm for the computation of Fourier integral operators, Multiscale Model. Simul. 7 (4) (2009) 1727–1750.
[3] B. Engquist, L. Ying, A fast directional algorithm for high frequency acoustic scattering in two dimensions, Commun. Math. Sci. 7 (2) (2009) 327–345.
[4] N. Halko, P.G. Martinsson, J.A. Tropp, Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions, SIAM Rev. 53 (2) (2011) 217–288.
[5] L. Hörmander, Fourier integral operators. I, Acta Math. 127 (1) (1971) 79–183.
[6] J. Hu, S. Fomel, L. Demanet, L. Ying, A fast butterfly algorithm for generalized Radon transforms, Geophysics 78 (4) (June 2013) U41–U51.
[7] Y. Li, H. Yang, E. Martin, K. Ho, L. Ying, Butterfly factorization, Multiscale Model. Simul. 13 (2) (2015) 714–732.
[8] Y. Li, H. Yang, L. Ying, A multiscale butterfly algorithm for multidimensional Fourier integral operators, Multiscale Model. Simul. 13 (2) (2015) 614–631.
[9] E. Liberty, F. Woolfe, P.-G. Martinsson, V. Rokhlin, M. Tygert, Randomized algorithms for the low-rank approximation of matrices, Proc. Natl. Acad. Sci. USA 104 (51) (2007) 20167–20172.

[10] L. Lin, J. Lu, L. Ying, Fast construction of hierarchical matrix representation from matrix–vector multiplication, J. Comput. Phys. 230 (10) (2011) 4071–4087.
[11] P.G. Martinsson, A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix, SIAM J. Matrix Anal. Appl. 32 (4) (2011) 1251–1274.
[12] P.G. Martinsson, V. Rokhlin, M. Tygert, A randomized algorithm for the decomposition of matrices, Appl. Comput. Harmon. Anal. 30 (1) (2011) 47–68.
[13] E. Michielssen, A. Boag, A multilevel matrix decomposition algorithm for analyzing scattering from large structures, IEEE Trans. Antennas and Propagation 44 (8) (Aug. 1996) 1086–1093.
[14] M. O'Neil, F. Woolfe, V. Rokhlin, An algorithm for the rapid evaluation of special function transforms, Appl. Comput. Harmon. Anal. 28 (2) (2010) 203–226.
[15] J. Poulson, L. Demanet, N. Maxwell, L. Ying, A parallel butterfly algorithm, SIAM J. Sci. Comput. 36 (1) (2014) C49–C65.
[16] D.S. Seljebotn, Wavemoth-fast spherical harmonic transforms by butterfly matrix compression, Astrophys. J., Suppl. Ser. 199 (1) (2012) 5.
[17] M. Tygert, Fast algorithms for spherical harmonic expansions, III, J. Comput. Phys. 229 (18) (2010) 6181–6192.
[18] F. Woolfe, E. Liberty, V. Rokhlin, M. Tygert, A fast randomized algorithm for the approximation of matrices, Appl. Comput. Harmon. Anal. 25 (3) (2008) 335–366.
[19] H. Yang, L. Ying, A fast algorithm for multilinear operators, Appl. Comput. Harmon. Anal. 33 (1) (2012) 148–158.
[20] L. Ying, Sparse Fourier transform via butterfly algorithm, SIAM J. Sci. Comput. 31 (3) (2009) 1678–1694.