BUTTERFLY FACTORIZATION VIA RANDOMIZED MATRIX-VECTOR MULTIPLICATIONS*

YANG LIU[§†], XIN XING[‡], HAN GUO[§], ERIC MICHIELSSEN[§], PIETER GHYSELS[†], AND XIAOYE SHERRY LI[†]

Abstract. This paper presents an adaptive randomized algorithm for computing the butterfly factorization of an $m \times n$ matrix with $m \approx n$ provided that both the matrix and its transpose can be rapidly applied to arbitrary vectors. The resulting factorization is composed of $\mathcal{O}(\log n)$ sparse factors, each containing $\mathcal{O}(n)$ nonzero entries. The factorization can be attained using $\mathcal{O}(n^{3/2} \log n)$ computation and $\mathcal{O}(n \log n)$ memory resources. The proposed algorithm can be implemented in parallel, and can apply to matrices with strong or weak admissibility conditions arising from surface integral equation solvers as well as multi-frontal-based finite-difference, finite-element, or finite-volume solvers. A distributed-memory parallel implementation of the algorithm demonstrates excellent scaling behavior.

Key word. Matrix factorization, butterfly algorithm, randomized algorithm, integral operator.

AMS subject classifications. 15A23, 65F50, 65R10, 65R20

1. Introduction.

Butterfly factorization is an important tool for compressing highly oscillatory operators arising in many scientific and engineering applications, such as the integral-equation-based analysis of high-frequency acoustic and electromagnetic scattering problems [25], the evaluation of Fourier integrals and transforms [2,33], spherical harmonic transforms [1,31], and other types of special function transforms [27]. The butterfly factorization of a $m \times n$ matrix with $m \approx n$ exists provided that all judiciously selected submatrices, whose row and column dimensions multiply to $\mathcal{O}(n)$, are numerically low-rank. Note that the submatrices can be non-contiguous if its rows and columns are not properly ordered. Through recursive low-rank factorizations of these submatrices, the operator can be represented as the product of $\mathcal{O}(\log n)$ sparse matrices, each containing $\mathcal{O}(n)$ nonzero entries. The resulting factorization can be rapidly applied to arbitrary vectors using only $\mathcal{O}(n \log n)$ computation and memory resources.

Despite this favorable application cost, the cost of constructing a butterfly representation of a given operator typically scales at least as $\mathcal{O}(n^2)$ [31]. Fortunately, there exist two important categories of operators that allow for fast approximation by a butterfly. (i) Operators that allow each element of their matrix representation to be evaluated in $\mathcal{O}(1)$ operations. This is typically the case when the butterfly factorization applies directly to an oscillatory operator with an explicit formula (e.g., Fourier operators, special transforms, or discretized integral equations) or stored as a full matrix. (ii) Operators (and their adjoints) with matrix representations that can be applied to arbitrary vectors in quasi-linear, typically $\mathcal{O}(n \log n)$, complexity. This situation typically arises when re-compressing the composition of highly-oscilatory operators, e.g., composition of Fourier integrals [12,14], matrix algebras for constructing discretized inverse integral operators [7]), compression of frontal matrices in multi-frontal sparse solvers [19], and conversion to a butterfly representation from other

*Version of November 11, 2020.

†Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA (liuyangzhuan@lbl.gov, pghysels@lbl.gov, xsli@lbl.gov,).

‡School of Mathematics, Georgia Institute of Technology, GA (xxing33@gatech.edu).

§Department of Electrical Engineering and Computer Science, University of Michigan, MI (hanguo@umich.edu, emichiel@umich.edu).

compression formats (e.g, fast multipole methods (FMM)-like formats). Before summarizing the key features of algorithms available in these categories, it is worth mentioning that this paper focuses on the development of fast butterfly algorithms for operators in category (ii).

The butterfly factorization of matrices in category (i) can be constructed using $\mathcal{O}(n \log n)$ computation and memory resources following the low-rank decomposition of judiciously selected submatrices using uniform [25], random [24], or Chebyshev [13, 28] proxy points. A wide variety of low-rank decompositions, including the interpolative decomposition (ID) [13, 28], the pseudo skeleton approximation [24], the adaptive cross approximation [30], and singular value decomposition (SVD) [14] can be used for this purpose. This type of butterfly factorization has been extended to multi-scale and multi-dimensional problems [15, 16, 25].

Operators in category (ii), on the other hand, post bigger challenges to fast butterfly construction algorithms. Existing algorithms [8, 14, 32] rely on random projection-based algorithms [9, 17] to construct low-rank decompositions of the associated submatrices, typically resulting in higher computation and memory costs than those in category (i). First, optimal-complexity (i.e., $\mathcal{O}(n \log n)$) algorithms exist when the oscillatory operator allows for smooth phase recovery [32] or fast submatrix-vector multiplications (e.g., using FMM-type algorithms). Unfortunately, these requirements are not met when compressing the concatenation of several Fourier operators or inverting integral equation operators. In addition, the iterative algorithm in [8] only exhibits rapid convergence for butterflies with a small number of levels. In comparison, the non-iterative algorithms in [8, 14] apply to butterfly compressible matrices with arbitrary levels, but require $\mathcal{O}(n^{3/2} \log n)$ computation resources due to the multiplication with multiple structured random matrices to address the specific submatrices, similar to the peeling algorithm in [18] for constructing $\mathcal{H}-$matrices. Specifically, the algorithm in [14] first constructs the innermost factor via randomized SVD and then moves towards the outermost factors using deterministic SVDs. This algorithm unfortunately requires $\mathcal{O}(n^{3/2})$ storage due to the need to store information associated with all random vectors. In contrast, the algorithm in [8] changes the computation sequence to an outside-in strategy and constructs every butterfly factor with randomized SVD. Despite having a slightly higher computational cost, this algorithm algorithm requires only $\mathcal{O}(n \log n)$ storage as it only stores information related to subsets of the structured random matrices.

This paper presents an improved butterfly reconstruction scheme based on the $\mathcal{O}(n^{3/2} \log n)$ computation and $\mathcal{O}(n \log n)$ memory algorithm of [8]: (i) The new algorithm leverages an improved adaptive scheme that permits fast and accurate butterfly reconstructions for matrices with non-constant butterfly ranks arising from the discretization of 3D surface integral equation solvers with weak admissibility. The algorithm in [8] expressly was designed for butterflies with constant rank, and exhibits higher computational costs when applied to cases with non-constant ranks. (ii) The algorithm comes with a rigorous error bound obtained using an orthogonal projection argument that grows only weakly with matrix size. The previous algorithm, in contrast, does not allow for the development of a rigorous bound as it uses random initial guesses for the butterfly factors to propagate the multiplication results towards the innermost factors. (iii) The algorithm can be deployed on distributed-memory computers. The previous algorithm, in contrast, is inherently sequential in nature, limiting its scalability when used in (even parallel) $\mathcal{H}-$matrix solvers [8]. The proposed algorithm represents a critical building block for constructing fast iterative and direct solvers for highly-oscillatory problems.

2. Preliminary background.

2.1. Notation. We use MATLAB notation to denote entries and subblocks of matrices and vectors. For example, $A(i,j)$ denotes the (i,j)th entry of matrix A, and $A(I,J)$ with index sets I and J denotes the subblock of matrix A with rows and columns indexed by I and J, respectively. We let $\mathrm{diag}(A_1,\ldots,A_k)$ denote a block diagonal matrix with blocks $A_1,\ldots,A_k$ on the diagonal. We always assume $A\in\mathbb{R}^{m\times n}$ and it is straightforward to extend all the following discussions to complex matrices.

2.2. Low-rank approximation by projection. Given a matrix $A\in\mathbb{R}^{m\times n}$, we consider a rank-$r$ approximation of A in the *projection form*,

$$A\approx UU^\top A,$$

where $U\in\mathbb{R}^{m\times r}$ has orthonormal columns and the symbol "$\top$" denotes the transpose of a matrix. The product $UU^\top$ projects all the columns of A onto the column space of U, denoted by $\mathrm{col}(U)$, which is of dimension r. Such a *basis matrix* U can be computed via SVD, pivoted QR, or randomized methods [17]. In this paper, we focus on using a typical randomized method illustrated in Algorithm 2.1 to obtain U and thus to construct a low-rank approximation by projection.

Algorithm 2.1 Randomized low-rank approximation method

Input: Matrix-matrix product routine of $A\in\mathbb{R}^{m\times n}$, relative error tolerance ϵ, initial rank guess r_0, over-sampling parameter p

Output: Basis matrix U for low-rank approximation $A\approx UU^\top A$

1: **Step 1:** Form a random matrix $\Omega\in\mathbb{R}^{n\times(r_0+p)}$ with its entries independently generated following the standard normal distribution.
2: **Step 2:** Compute $W=A\Omega\in\mathbb{R}^{m\times(r_0+p)}$.
3: **Step 3:** Compute the column-pivoted QR decomposition of W as $WP=QR$ where $Q\in\mathbb{R}^{m\times(r_0+p)}$ is orthonormal, and $R\in\mathbb{R}^{(r_0+p)\times(r_0+p)}$ is upper triangular. Truncate the QR decomposition with relative error tolerance ϵ, i.e., find the maximum index r satisfying that $|R(r,r)|>\epsilon|R(1,1)|$. Return U consisting of the first r columns of Q.

This randomized method is extensively studied in [5,9]. Theoretically, it is shown that with high probability the rank-r approximation $UU^\top A$ constructed by Algorithm 2.1 has nearly optimal approximation error (see Theorem 10.7 of [9]). Experimentally, this approximation usually can have actual relative error of similar scale as the specified relative error tolerance ϵ in Algorithm 2.1, i.e., $\|A-UU^\top A\|_F/\|A\|_F\sim\mathcal{O}(\epsilon)$. Algorithm 2.1 has $\mathcal{O}(mnr)$ computation cost when A is a full matrix and could be more efficient if any fast matrix-matrix product algorithm for A is available.

3. Butterfly factorization.

We consider the butterfly factorization of a special matrix $K(T,S)$ defined by a highly-oscillatory operator $K(\cdot,\cdot)$ and point sets S and T. For example, consider the free-space wave interactions between 3D *source* points S and *target* points T where S and T are non-overlapping (weak admissibility) or well separated (strong admissibility). Matrix $K(T,S)$ consists of entries $K(t_i,s_j)=\exp(i2\pi\kappa|t_i-s_j|)/|t_i-s_j|$ for all pairs $(t_i,s_j)\in T\times S$ and is a discretization of the 3D Helmholtz problem with wave number $\kappa>0$. Let $|T|=m$ and $|S|=n$, and assume that $m=\mathcal{O}(n)$. Other examples of highly-oscillatory operators includes Green's function operators for Helmholtz equations with non-constant coefficients, Fourier transforms, and special function transforms.

3.1. Hierarchical partitioning. The point sets S and T are both partitioned recursively into small subsets until each finest subset has the number of points inside less than a prescribed small constant n_0. Such a partitioning of S/T is characterized by a partition tree whose each node corresponds to one subset. For simplicity, we consider bisection of the sets and denote the resulting two binary partition trees for S and T as $\mathcal{T}_S$ and $\mathcal{T}_T$, respectively. We further assume both $\mathcal{T}_S$ and $\mathcal{T}_T$ are perfect (all levels are filled completely). Note that these assumptions can be easily lifted.

Let T_τ be the subset of points in T corresponding to node τ in $\mathcal{T}_T$. For the root node t of $\mathcal{T}_T$, $T_t = T$; for each nonleaf node $\tau \in \mathcal{T}_T$ with children τ_1 and τ_2, $T_{\tau_1} \cup T_{\tau_2} = T_\tau$ and $T_{\tau_1} \cap T_{\tau_2} = \emptyset$. With slight abuse of notation, we also use $\tau_i,\ i = 1, \ldots, 2^l$ to denote all nodes at level l of $\mathcal{T}_T$. The same applies to the hierarchical partitioning of S, i.e., $\{S_\nu\}_{\nu \in \mathcal{T}_S}$. We assume that both $\mathcal{T}_T$ and $\mathcal{T}_S$ have the same depth $L = \mathcal{O}(\log n)$ so that each subset associated with a leaf node has $\mathcal{O}(1)$ points. With such hierarchical partitioning of T and S, the points in T and S can be properly reordered so that points in each subset T_τ/S_ν correspond to consecutive rows/columns in $K(T, S)$.

We number the levels of $\mathcal{T}_T$ and $\mathcal{T}_S$ from the root to the leafs. The root node is at level 0; its children are at level 1, etc. All the leaf nodes are at level L. At each level l, $\mathcal{T}_T$ and $\mathcal{T}_S$ both have 2^l nodes.

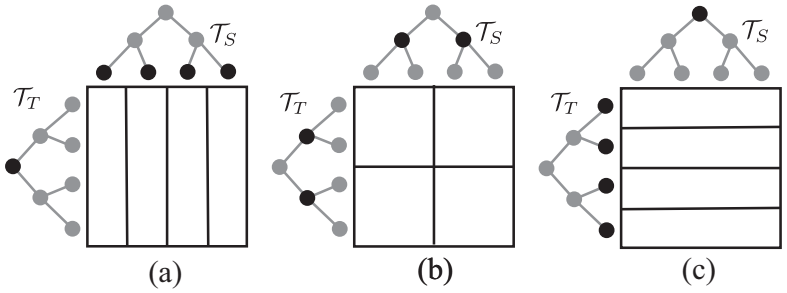

FIG. 3.1. *The partition trees $\mathcal{T}_S$ and $\mathcal{T}_T$, and the blocks $K(T_\tau, S_\nu)$ for a 2-level butterfly factorization. These blocks correspond to level l of $\mathcal{T}_T$ and level $L - l$ of $\mathcal{T}_S$ with (a) $l = 0$ (b) $l = 1$ (c) $l = 2$.*

3.2. Complementary low-rank property. It turns out that matrix $K(T, S)$ satisfies the *complementary low-rank property* that for any level $0 \le l \le L$, node τ at level l of $\mathcal{T}_T$ and node ν at level $(L - l)$ of $\mathcal{T}_S$, the subblock $K(T_\tau, S_\nu)$ has its numerical rank bounded by some small constant r. This constant r is referred to as the butterfly rank. Figure 3.1 illustrates these subblocks with a two-level partitioning of T and S. Such complementary low-rank property can result from certain analytic properties of Helmholtz kernel [4, 21, 24], Fourier transform operator [2], and many other highly-oscillatory operators.

3.3. Low-rank approximation of blocks. A butterfly factorization of $K(T, S)$ compresses all blocks $K(T_\tau, S_\nu)$ with $l = 0, 1, \ldots, L$ for nodes τ at level l of $\mathcal{T}_T$ and ν at level $(L - l)$ of $\mathcal{T}_S$ (referred to as the blocks at level l of the butterfly factorization) into low-rank form via a nested approach. There are three similar forms: (1) column-wise butterfly factorization, (2) row-wise butterfly factorization [2, 8, 24, 27, 31, 33], and (3) hybrid butterfly factorization [1, 13, 14, 28]. Our proposed algorithm is based on the hybrid factorization as it leads to the lowest algorithmic complexity and the highest parallel efficiency among the three.

In the following, we describe all the three forms as the column-wise and row-wise factorizations also serve as building blocks for the hybrid factorization. For brevity, we assume all blocks $K(T_\tau, S_\nu)$ are compressed into rank-r form.

3.3.1. Column-wise butterfly factorization. Each $K(T_\tau, S_\nu)$ is compressed into rank-r form

$$K(T_\tau, S_\nu) \approx U_{\tau,\nu} U_{\tau,\nu}^\top K(T_\tau, S_\nu) = U_{\tau,\nu} E_{\tau,\nu}, \tag{3.1}$$

where $U_{\tau,\nu} \in \mathbb{R}^{|T_\tau| \times r}$ has orthonormal columns and is referred to as *the column basis matrix* associated with (τ, ν).

Before considering an arbitrary butterfly level L, we first illustrate the column-wise factorization using a simple example with $L = 1$. Let t and s denote the root nodes of $\mathcal{T}_T$ and $\mathcal{T}_S$; let τ_1, τ_2 and ν_1, ν_2 denote the leaf nodes of $\mathcal{T}_T$ and $\mathcal{T}_S$. By the complementary low-rank property, $K(T_{\tau_1}, S_s)$, $K(T_{\tau_2}, S_s)$, $K(T_t, S_{\nu_1})$, and $K(T_t, S_{\nu_2})$ have numerical ranks at most r. First, we construct the low-rank approximations $K(T_{\tau_1}, S_s) \approx U_{\tau_1,s} E_{\tau_1,s}$ and $K(T_{\tau_2}, S_s) \approx U_{\tau_2,s} E_{\tau_2,s}$. Next, we split each $E_{\tau_i,s}$, $i = 1, 2$ by columns into $[E_{\tau_i,\nu_1},\ E_{\tau_i,\nu_2}]$ and consider two vertical concatenations $[E_{\tau_1,\nu_j}; E_{\tau_2,\nu_j}]$, $j = 1, 2$. Since $[E_{\tau_1,\nu_j}; E_{\tau_2,\nu_j}]$ has the same row span as $K(T_t, S_{\nu_j})$, it can be approximated by a rank-r form $[E_{\tau_1,\nu_j}; E_{\tau_2,\nu_j}] \approx R_{t,\nu_j} E_{t,\nu_j}$ where R_{t,ν_j} has orthonormal columns. The overall process above can be expressed as,

$$\begin{aligned}
K(T_t, S_s) &= \begin{bmatrix} K(T_{\tau_1}, S_s) \\ K(T_{\tau_2}, S_s) \end{bmatrix} \approx \begin{bmatrix} U_{\tau_1,s} & \\ & U_{\tau_2,s} \end{bmatrix} \begin{bmatrix} E_{\tau_1,s} \\ E_{\tau_2,s} \end{bmatrix} \\
&= \begin{bmatrix} U_{\tau_1,s} & \\ & U_{\tau_2,s} \end{bmatrix} \left[\begin{bmatrix} E_{\tau_1,\nu_1} \\ E_{\tau_2,\nu_1} \end{bmatrix} \quad \begin{bmatrix} E_{\tau_1,\nu_2} \\ E_{\tau_2,\nu_2} \end{bmatrix} \right] \\
&\approx \begin{bmatrix} U_{\tau_1,s} & \\ & U_{\tau_2,s} \end{bmatrix} \begin{bmatrix} R_{t,\nu_1} & R_{t,\nu_2} \end{bmatrix} \begin{bmatrix} E_{t,\nu_1} & \\ & E_{t,\nu_2} \end{bmatrix}.
\end{aligned}$$

The last equation gives a one-level column-wise butterfly factorization of $K(T_t, S_s)$.

In general for $L \geq 1$, the column-wise factorization proceeds as follows: At the leaf level L of $\mathcal{T}_T$, the basis matrix $U_{\tau,\nu}$ is explicitly formed for each T_τ. At a non-leaf level $l < L$, consider a node τ at level l of $\mathcal{T}_T$ and a node ν at level $(L - l)$ of $\mathcal{T}_S$. Let $\{\tau_1, \tau_2\}$ be the children of τ at level $(l + 1)$ of $\mathcal{T}_T$ and p_ν be the parent node of ν at level $(L - l - 1)$ of $\mathcal{T}_S$. The low-rank approximation (3.1) of $K(T_\tau, S_\nu)$ is constructed by exploiting the available low-rank approximations of blocks $K(T_{\tau_1}, S_{p_\nu})$ and $K(T_{\tau_2}, S_{p_\nu})$ at level $(l + 1)$ in the following nested manner.

Since $T_\tau = T_{\tau_1} \cup T_{\tau_2}$, $K(T_\tau, S_\nu)$ can first be split into two blocks,

$$K(T_\tau, S_\nu) = \begin{bmatrix} K(T_{\tau_1}, S_\nu) \\ K(T_{\tau_2}, S_\nu) \end{bmatrix}. \tag{3.2}$$

Meanwhile, since S_ν is a subset of S_{p_ν}, it follows that $K(T_{\tau_a}, S_\nu)$, for each child τ_a of τ, is a subblock of $K(T_{\tau_a}, S_{p_\nu})$. Thus, the low-rank approximation $K(T_{\tau_a}, S_{p_\nu}) \approx U_{\tau_a,p_\nu} E_{\tau_a,p_\nu}$ at level $(l + 1)$ yields a low-rank approximation of $K(T_{\tau_a}, S_\nu)$ as,

$$K(T_{\tau_a}, S_\nu) \approx U_{\tau_a,p_\nu} E_{\tau_a,\nu},$$

where $E_{\tau_a,\nu}$ is a subset of columns in E_{τ_a,p_ν} corresponding to S_ν. Substituting this approximation into (3.2) gives

$$K(T_\tau, S_\nu) \approx \begin{bmatrix} U_{\tau_1,p_\nu} E_{\tau_1,\nu} \\ U_{\tau_2,p_\nu} E_{\tau_2,\nu} \end{bmatrix} = \begin{bmatrix} U_{\tau_1,p_\nu} & \\ & U_{\tau_2,p_\nu} \end{bmatrix} \begin{bmatrix} E_{\tau_1,\nu} \\ E_{\tau_2,\nu} \end{bmatrix}. \tag{3.3}$$

Instead of directly compressing $K(T_\tau, S_\nu)$, we compute a rank-r approximation of the last matrix above, which only has $2r$ rows and is far smaller than $K(T_\tau, S_\nu)$, as

$$\begin{bmatrix} E_{\tau_1,\nu} \\ E_{\tau_2,\nu} \end{bmatrix} \approx R_{\tau,\nu} E_{\tau,\nu}, \tag{3.4}$$

where $R_{\tau,\nu}$ has orthonormal columns. Substituting (3.4) into (3.3), we obtain a rank-r approximation of $K(T_\tau, S_\nu)$ as,

$$K(T_\tau, S_\nu) \approx \begin{bmatrix} U_{\tau_1,p_\nu} & \\ & U_{\tau_2,p_\nu} \end{bmatrix} R_{\tau,\nu} E_{\tau,\nu} = U_{\tau,\nu} E_{\tau,\nu},$$

where the column basis matrix $U_{\tau,\nu}$ is

$$U_{\tau,\nu} = \begin{bmatrix} U_{\tau_1,p_\nu} & \\ & U_{\tau_2,p_\nu} \end{bmatrix} R_{\tau,\nu}, \tag{3.5}$$

and $R_{\tau,\nu}$ is referred to as a *transfer matrix*; note that $U_{\tau,\nu}$ still has orthonormal columns.

Using (3.5), the basis matrices at any non-leaf level l are expressed in terms of the basis matrices at level $(l+1)$ via the transfer matrices. Thus, the basis matrices at any non-leaf level are not explicitly formed but instead recovered recursively from quantities at lower levels. In the end, the butterfly factorization of $K(T, S)$ consists of the low-rank approximations of blocks at level 0 of $\mathcal{T}_T$, i.e.,

$$\begin{aligned} K(T,S) &= \begin{bmatrix} K(T_t, S_{\nu_1}) & K(T_t, S_{\nu_2}) & \dots & K(T_t, S_{\nu_{2^L}}) \end{bmatrix} \\ &\approx \begin{bmatrix} U_{t,\nu_1} E_{t,\nu_1} & U_{t,\nu_2} E_{t,\nu_2} & \dots & U_{t,\nu_{2^L}} E_{t,\nu_{2^L}} \end{bmatrix} && (3.6) \\ &= \left(U^L R^{L-1} R^{L-2} \dots R^0\right) E^0, && (3.7) \end{aligned}$$

where t denotes the root node of $\mathcal{T}_T$, $\nu_1, \dots, \nu_{2^L}$ are all the leaf nodes of $\mathcal{T}_S$, and $U_{t,\nu_1}, \dots, U_{t,\nu_{2^L}}$ are the corresponding column basis matrices. Expanding each U_{t,ν_a} using the nested form (3.5) up to the leaf level of $\mathcal{T}_T$, $K(T, S)$ can be represented as the product of the $(L+2)$ matrices in (3.7) where $U^L = \mathrm{diag}(U_{\tau_1,s}, \dots, U_{\tau_{2^L},s})$ consists of all the column basis matrices at level L of $\mathcal{T}_T$ (s is the root of $\mathcal{T}_S$), $E^0 = \mathrm{diag}(E_{t,\nu_1}, \dots, E_{t,\nu_{2^L}})$, and each factor $R^l, l = 0, \dots, L-1$ is a block diagonal matrix consisting of special blocks R_ν for all nodes ν at level l of $\mathcal{T}_S$. Here, each R_ν consists of R_{τ,ν_1} and R_{τ,ν_2} for all nodes τ at level $L-l$ of $\mathcal{T}_T$ as

$$R_\nu = \begin{bmatrix} R_{\tau_1,\nu_1} & & R_{\tau_1,\nu_2} & \\ & R_{\tau_2,\nu_1} & & R_{\tau_2,\nu_2} \\ \ddots & & \ddots & \\ & R_{\tau_{2^{L-l}},\nu_1} & & R_{\tau_{2^{L-l}},\nu_2} \end{bmatrix} \tag{3.8}$$

where $\{\nu_1, \nu_2\}$ denotes the children of ν. We term U^L and E^0 *outer factors* and R^l *inner factors*. Figure 3.2(a) shows an example of a 4-level column-wise factorization.

3.3.2. Row-wise butterfly factorization. Each $K(T_\tau, S_\nu)$ is compressed into rank-r form as

$$K(T_\tau, S_\nu) \approx K(T_\tau, S_\nu) V_{\tau,\nu} V_{\tau,\nu}^\top = F_{\tau,\nu} V_{\tau,\nu}^\top, \tag{3.9}$$

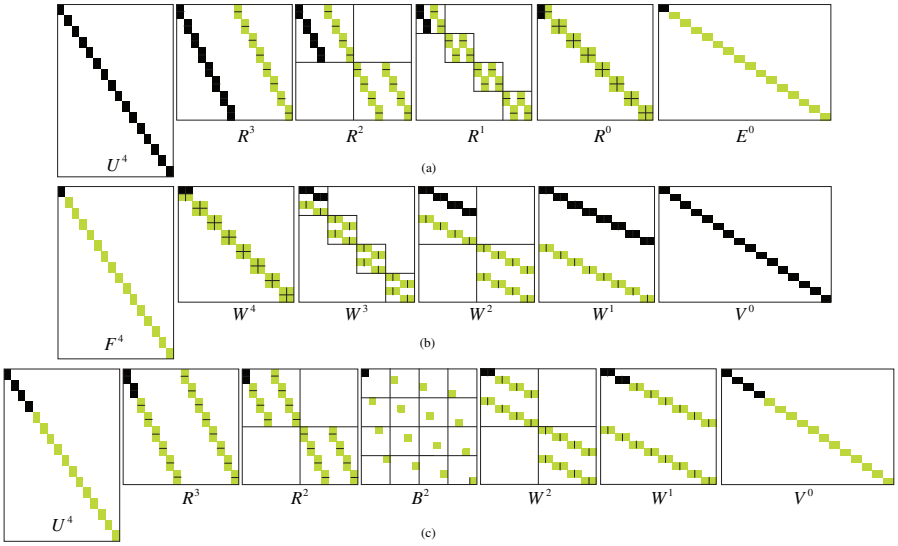

FIG. 3.2. *(a) Column-wise, (b) row-wise, and (c) hybrid butterfly factorizations for a 4-level hierarchical partitioning of T and S. The blocks in black multiply to $U_{t,\nu_1}E_{t,\nu_1}$ in (3.6), $F_{\tau_1,s}V_{\tau_1,s}^\top$, and $U_{\tau_1,\nu_1}B_{\tau_1,\nu_1}V_{\tau_1,\nu_1}^\top$ in (3.13), respectively.*

where $V_{\tau,\nu} \in \mathbb{R}^{|S_\nu|\times r}$ has orthonormal columns and is referred to as *the row basis matrix* associated with (τ,ν). Just like for the column-wise factorization, we define the transfer matrix $W_{\tau,\nu}$ for a non-leaf node ν as

$$V_{\tau,\nu} = \begin{bmatrix} V_{p_\tau,\nu_1} & \\ & V_{p_\tau,\nu_2} \end{bmatrix} W_{\tau,\nu}. \tag{3.10}$$

The basis and transfer matrices can be constructed upon applying the column-wise butterfly factorization to $K(T,S)^\top$, yielding the row-wise butterfly structure

$$K(T,S) = F^L\big(W^L W^{L-1}\dots W^1 V^0\big). \tag{3.11}$$

The outer factors are $F^L = \mathrm{diag}(F_{\tau_1,s},\dots,F_{\tau_{2^L},s})$ with s denoting the root of $\mathcal{T}_S$ and $V^0 = \mathrm{diag}(V_{t,\nu_1}^\top,\dots,V_{t,\nu_{2^L}}^\top)$. The block diagonal inner factors $W^l, l=1,\dots,L$ consist of blocks W_τ for all nodes τ at level $l-1$ of $\mathcal{T}_T$. Each W_τ consists of $W_{\tau_1,\nu}$ and $W_{\tau_2,\nu}$ for all nodes ν at level $L-l+1$ of $\mathcal{T}_S$, as

$$W_\tau = \begin{bmatrix} W_{\tau_1,\nu_1} & & & \\ & W_{\tau_1,\nu_2} & & \\ & & \ddots & \\ & & & W_{\tau_1,\nu_{2^{L-l+1}}} \\ W_{\tau_2,\nu_1} & & & \\ & W_{\tau_2,\nu_2} & & \\ & & \ddots & \\ & & & W_{\tau_2,\nu_{2^{L-l+1}}} \end{bmatrix}. \tag{3.12}$$

Figure 3.2(b) shows an example of a 4-level row-wise factorization.

3.3.3. Hybrid butterfly factorization. At any level l of $\mathcal{T}_T$, $K(T_\tau, S_\nu)$ with all nodes τ at level l of $\mathcal{T}_T$ and nodes ν at level $(L-l)$ of $\mathcal{T}_S$ form a non-overlapping partitioning of $K(T,S)$. Fixing l, we can combine the computed row and column basis matrices $U_{\tau,\nu}$, $V_{\tau,\nu}$ from both the column-wise and row-wise butterfly factorizations of $K(T,S)$ above to compress $K(T_\tau, S_\nu)$ as

$$K(T_\tau, S_\nu) \approx U_{\tau,\nu} U_{\tau,\nu}^\top K(T_\tau, S_\nu) V_{\tau,\nu} V_{\tau,\nu}^\top = U_{\tau,\nu} B_{\tau,\nu} V_{\tau,\nu}^\top.$$

The hybrid butterfly factorization of $K(T,S)$ is constructed as,

$$K(T,S) = \begin{bmatrix} K(T_{\tau_1}, S_{\nu_1}) & K(T_{\tau_1}, S_{\nu_2}) & \cdots & K(T_{\tau_1}, S_{\nu_q}) \\ K(T_{\tau_2}, S_{\nu_1}) & K(T_{\tau_2}, S_{\nu_2}) & \cdots & K(T_{\tau_2}, S_{\nu_q}) \\ \vdots & \vdots & \ddots & \vdots \\ K(T_{\tau_p}, S_{\nu_1}) & K(T_{\tau_p}, S_{\nu_2}) & \cdots & K(T_{\tau_p}, S_{\nu_q}) \end{bmatrix}$$

$$\approx \begin{bmatrix} U_{\tau_1,\nu_1} B_{\tau_1,\nu_1} V_{\tau_1,\nu_1}^\top & U_{\tau_1,\nu_2} B_{\tau_1,\nu_2} V_{\tau_1,\nu_2}^\top & \cdots & U_{\tau_1,\nu_q} B_{\tau_1,\nu_q} V_{\tau_1,\nu_q}^\top \\ U_{\tau_2,\nu_1} B_{\tau_2,\nu_1} V_{\tau_2,\nu_1}^\top & U_{\tau_2,\nu_2} B_{\tau_2,\nu_2} V_{\tau_2,\nu_2}^\top & \cdots & U_{\tau_2,\nu_q} B_{\tau_2,\nu_q} V_{\tau_2,\nu_q}^\top \\ \vdots & \vdots & \ddots & \vdots \\ U_{\tau_p,\nu_1} B_{\tau_p,\nu_1} V_{\tau_p,\nu_1}^\top & U_{\tau_p,\nu_2} B_{\tau_p,\nu_2} V_{\tau_p,\nu_2}^\top & \cdots & U_{\tau_p,\nu_q} B_{\tau_p,\nu_q} V_{\tau_p,\nu_q}^\top \end{bmatrix} \tag{3.13}$$

$$= \big(U^L R^{L-1} R^{L-2} \dots R^l\big) B^l \big(W^l W^{l-1} \dots W^1 V^0\big) \tag{3.14}$$

where $\tau_1, \tau_2, \dots, \tau_p$ are the $p = 2^l$ nodes at level l of $\mathcal{T}_T$, and $\nu_1, \nu_2, \dots, \nu_q$ are the $q = 2^{L-l}$ nodes at level $(L-l)$ of $\mathcal{T}_S$.

In this level-l hybrid butterfly factorization, these column basis matrices $U_{\tau,\nu}$ are recursively defined as in (3.5) using column basis and transfer matrices in the lower levels of $\mathcal{T}_T$, and the row basis matrices $V_{\tau,\nu}$ are recursively defined as in (3.10) using row basis and transfer matrices at upper levels of $\mathcal{T}_T$. In (3.14), the outer factors U^L, V^0 and inner factors $\{R^k\}, \{W^k\}$ are defined in Sections 3.3.1 and 3.3.2, and the inner factor B^l consists of all blocks $B_{\tau,\nu}$ at level l in (3.13). For simplicity assuming $r_{\tau,\nu} = r$, B^l is a $p \times q$ block-partitioned matrix with each block of sizes $qr \times pr$; the (i,j) block is a $q \times p$ block-partitioned matrix with each block of sizes $r \times r$, among which the only nonzero block is the (j,i) block and equals B_{τ_i,ν_j}. Typically, the level l is set to center level $l_c = \lfloor L/2 \rfloor$. Figure 3.2(c) shows a hybrid butterfly factorization with $L = 4$ and $l_c = 2$.

4. Adaptive butterfly factorization via randomized matrix-vector products. We propose an algorithm for constructing the butterfly factorization of a matrix $A = K(T,S)$ using only products of A and its transpose with random vectors. The proposed Algorithm 4.1 returns a hybrid factorization with prescribed tolerance ϵ assuming black-box matrix-vector multiplications. With minor modifications, Algorithm 4.1 also applies to column- and row-wise factorizations albeit with a much higher computational cost, i.e., $\mathcal{O}(n^2 \log n)$. In what follows, we describe the four key components of the algorithm, including

- The computation of $K(T_\tau, S_\nu)\Omega_\nu$ and $K(T_\tau, S_\nu)^\top \Gamma_\tau$ with random matrices Ω_ν and Γ_τ using a black-box routine.
- The construction of column basis matrices $U_{\tau,\nu}$ (or transfer matrices $R_{\tau,\nu}$ for non-leaf nodes τ in $\mathcal{T}_T$) based on matrix-vector multiplications involving $K(T,S)$.
- The construction of row basis matrices $V_{\tau,\nu}$ (or transfer matrices $W_{\tau,\nu}$ for non-leaf nodes ν in $\mathcal{T}_S$) based on the matrix-vector multiplications involving $K(T,S)^\top$.

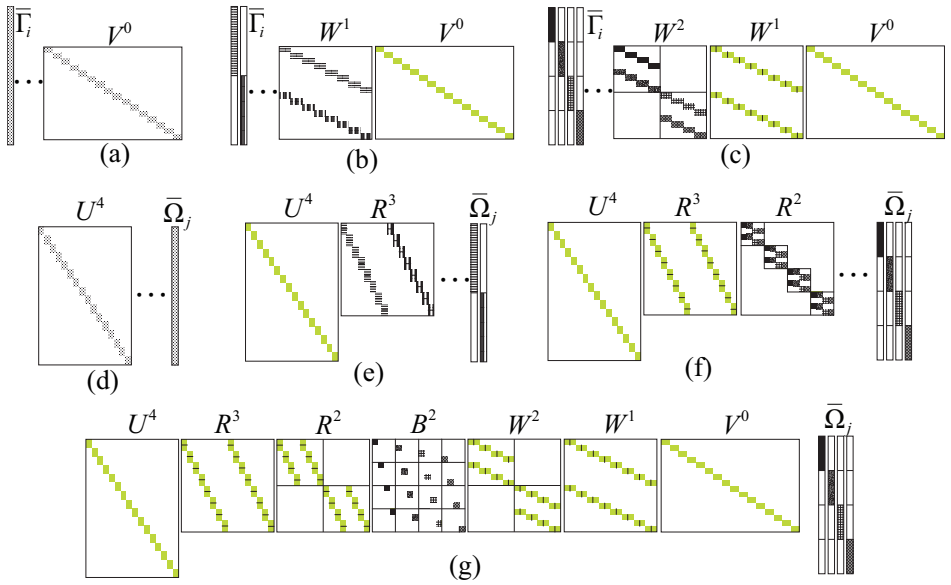

FIG. 3.3. *A 4-level hybrid factorization based on matrix-vector products consists of steps that compute (a) V^0, (b) W^0, (c) W^1, (d) U^4, (e) R^3, (f) R^2, and (g) B^2. Note that the vectors $\bar{\Omega}_\nu$ or $\bar{\Gamma}_\tau$ and the blocks being computed are marked with the same texture. The already-computed blocks needed at each step are plotted in Green.*

- The construction of intermediate matrices B_{τ_a,ν_b} in (3.13).

4.1. Multiplication of $K(T_\tau, S_\nu)$ and $K(T_\tau, S_\nu)^\top$ by random matrices. Here we assume the existence of a black-box program to perform matrix-vector multiplications involving $K(T, S)$ and $K(T, S)^\top$. To use Algorithm 2.1 to compute the column/row basis matrices for each block $K(T_\tau, S_\nu)$, we multiply $K(T, S)$ or $K(T, S)^\top$ with *structured random matrices* $\bar{\Omega}_\nu$ and $\bar{\Gamma}_\tau$ to obtain the matrices $K(T_\tau, S_\nu)\Omega_\nu$ and $K(T_\tau, S_\nu)^\top\Gamma_\tau$ that are used in Algorithm 2.1. As we shall see later, Ω_ν and Γ_τ are sub-vectors of $\bar{\Omega}_\nu$ and $\bar{\Gamma}_\tau$. Their entries are random variables which are independent and identically distributed, following a normal distribution.

Fixing a node ν at level $(L-l)$ of $\mathcal{T}_S$, we compute $K(T_\tau, S_\nu)\Omega_\nu$ with a $|S_\nu| \times (r+p)$ matrix Ω_ν for all nodes τ at level l of $\mathcal{T}_T$, by multiplying $K(T, S)$ with a sparse $|S| \times (r + p)$ matrix $\bar{\Omega}_\nu$ whose only non-zero entries Ω_ν are located on the rows corresponding to S_ν. To evaluate all the multiplications $K(T_\tau, S_\nu)\Omega_\nu$ at level l (of $\mathcal{T}_T$), $2^{(L-l)}(r + p)$ matrix-vector multiplications by $K(T, S)$ are needed.

Similarly, fixing a node τ at level l of $\mathcal{T}_T$, we compute $K(T_\tau, S_\nu)^\top\Gamma_\tau$ with a $|T_\tau| \times (r+p)$ matrix Γ_τ for all nodes ν at level $(L-l)$ of $\mathcal{T}_S$, by multiplying $K(T, S)^\top$ with a sparse matrix $\bar{\Gamma}_\tau \in \mathbb{R}^{|T|\times(r+p)}$ whose only non-zero entries Γ_τ are located on the rows corresponding to T_τ. To evaluate all the multiplications $K(T_\tau, S_\nu)^\top\Gamma_\tau$ at level l (of $\mathcal{T}_T$), $2^l(r + p)$ matrix-vector multiplications by $K(T, S)^\top$ are needed. In Algorithm 4.1, the products $K(T_\tau, S_\nu)\Omega_\nu$ and $K(T_\tau, S_\nu)^\top\Gamma_\tau$ are performed on lines 4, 23, 13, 33.

4.2. Computation of $U_{\tau,\nu}$ and $R_{\tau,\nu}$. For each leaf node τ at level L of $\mathcal{T}_T$ and the root node s of $\mathcal{T}_S$, the column basis matrix $U_{\tau,s}$ can be directly computed using Algorithm 2.1 by evaluating $K(T_\tau, S_s)\Omega_s$. In Algorithm 4.1 (line 2), the ranks of $U_{\tau,s}$ are determined by adaptively doubling the size of the random matrices Ω_s for

Algorithm 4.1 Adaptive and randomized hybrid butterfly factorization based on matrix-vector multiplication

Input: Black-box routine for multiplying $K(T,S) \in \mathbb{R}^{m\times n}$ and its transpose with arbitrary matrices, over-sampling parameter p, relative error tolerance ϵ, initial rank guess r_0, binary partitioning trees $\mathcal{T}_S$ and $\mathcal{T}_T$ of L levels.

Output: $K(T,S) \approx (U^L R^{L-1} R^{L-2} \dots R^{l_c}) B^{l_c} (W^{l_c} W^{l_c-1} \dots W^1 V^0)$ with $l_c = \lfloor L/2 \rfloor$.

$r = r_0$.
while not converged **do** ▷ Adaptive computation of $V_{\tau,\nu}$ at level 0
  Form a random matrix $\Gamma_t \in \mathbb{R}^{|T_t|\times(r+p)}$ for the root node t of $\mathcal{T}_T$.
  Compute $K(T_t,S)^\top \Gamma_t$.
  for ν at level L of $\mathcal{T}_S$ **do**
    Apply Algorithm 2.1 with $A = K(T_t,S_\nu)^\top$ to compute $V_{t,\nu}$.
  end for
  Converge if $r > \max_\nu\{r_{t,\nu}\}$. ▷ Over all nodes ν at leaf level of $\mathcal{T}_S$
  $r \leftarrow 2r$.
end while
while not converged **do** ▷ Adaptive computation of $U_{\tau,\nu}$ at level L
  Form a random matrix $\Omega_s \in \mathbb{R}^{|S_s|\times(r+p)}$ for the root node s of $\mathcal{T}_S$.
  Compute $K(T,S_s)\Omega_s$.
  for τ at level L of $\mathcal{T}_T$ **do**
    Apply Algorithm 2.1 with $A = K(T_\tau,S_s)$ to compute $U_{\tau,s}$.
  end for
  Converge if $r > \max_\tau\{r_{\tau,s}\}$. ▷ Over all nodes τ at leaf level of $\mathcal{T}_T$
  $r \leftarrow 2r$.
end while
for $l = 1$ to l_c **do** ▷ Computation of $W_{\tau,\nu}$
  for τ at level l of $\mathcal{T}_T$ **do**
    Form $\Gamma_\tau \in \mathbb{R}^{|T_\tau|\times(r+p)}$ with $r = \max_\nu\{r_{p_\tau,\nu_1} + r_{p_\tau,\nu_2}\}$. ▷ Over all nodes ν at level $L-l$ of $\mathcal{T}_S$
    Compute $K(T_\tau,S)^\top\Gamma_\tau$.
    for ν at level $L-l$ of $\mathcal{T}_S$ **do**
      Compute $\begin{bmatrix} V_{p_\tau,\nu_1}^\top & \\ & V_{p_\tau,\nu_2}^\top \end{bmatrix} (K(T_\tau,S_\nu)^\top\Gamma_\tau) = A\Gamma_\tau$
      Apply Algorithm 2.1 for $W_{\tau,\nu}$ with A above.
    end for
  end for
end for
for $l = L-1$ to l_c **do** ▷ Computation of $R_{\tau,\nu}$ and $B_{\tau,\nu}$
  for ν at level $L-l$ of $\mathcal{T}_S$ **do**
    Form $\Omega_\nu \in \mathbb{R}^{|S_\nu|\times(r+p)}$ with $r = \max_\tau\{r_{\tau_1,p_\nu} + r_{\tau_2,p_\nu}\}$. ▷ Over all nodes τ at level l of $\mathcal{T}_T$
    Compute $K(T,S_\nu)\Omega_\nu$.
    Compute $V_{\tau,\nu}^\top\Omega_\nu$ if $l = l_c$. ▷ $V_{\tau,\nu}^\top$ is not explicitly computed.
    for τ at level $L-l$ of $\mathcal{T}_T$ **do**
      Compute $\begin{bmatrix} U_{\tau_1,p_\nu}^\top & \\ & U_{\tau_2,p_\nu}^\top \end{bmatrix} (K(T_\tau,S_\nu)\Omega_\nu) = A\Omega_\nu$
      Apply Algorithm 2.1 for $R_{\tau,\nu}$ with A above.
      Compute $B_{\tau,\nu}$ using (4.1) if $l = l_c$.
    end for
  end for
end for

$U_{\tau,s}$ until convergence. The iteration is terminated if the rank estimate r exceeds the maximum revealed rank $\max_{\nu}\{r_{t,\nu}\}$ (see line 17). This heuristic stopping criterion is used as a more rigorous criterion requires expensive computation of the approximation error.

For each non-leaf node τ at level $l_c \le l < L$ of $\mathcal{T}_T$ and each node ν at level $(L-l)$ of $\mathcal{T}_S$, the matrix to be compressed in (3.3) when computing $R_{\tau,\nu}$ can be expressed as,

$$\begin{bmatrix} E_{\tau_1,\nu} \\ E_{\tau_2,\nu} \end{bmatrix} = \begin{bmatrix} U_{\tau_1,p_\nu}^\top & \\ & U_{\tau_2,p_\nu}^\top \end{bmatrix} K(T_\tau, S_\nu) \approx R_{\tau,\nu} E_{\tau,\nu}.$$

Thus, $R_{\tau,\nu}$ can be computed via Algorithm 2.1 using the products

$$\begin{bmatrix} U_{\tau_1,p_\nu}^\top & \\ & U_{\tau_2,p_\nu}^\top \end{bmatrix} K(T_\tau, S_\nu)\Omega_\nu.$$

Note that no rank adaptation is needed for $R_{\tau,\nu}$ in Algorithm 4.1 as the rank $r_{\tau,\nu}$ is bounded by $r_{\tau_1,p_\nu} + r_{\tau_2,p_\nu}$. The dimensions of the random matrices Ω_ν therefore are chosen using the rank estimate $r = \max_{\nu}\{r_{\tau_1,p_\nu} + r_{\tau_2,p_\nu}\}$ on line 32. This process recursively traverses $\mathcal{T}_T$ from the leafs to center level l_c and $\mathcal{T}_S$ from the root to center level l_c.

4.3. Computation of $V_{\tau,\nu}$ and $W_{\tau,\nu}$. The computation of $V_{\tau,\nu}$ and $W_{\tau,\nu}$ resembles the above computation of $U_{\tau,\nu}$ and $R_{\tau,\nu}$, but uses the multiplication results $K(T_\tau, S_\nu)^\top \Gamma_\tau$. $V_{\tau,\nu}$ for leaf nodes is computed adaptively on line 11 while $W_{\tau,\nu}$ for non-leaf nodes is computed using Algorithm 2.1 using the multiplication results

$$\begin{bmatrix} V_{p_\tau,\nu_1}^\top & \\ & V_{p_\tau,\nu_2}^\top \end{bmatrix} K(T_\tau, S_\nu)^\top \Gamma_\tau.$$

where the dimensions of the random matrices Γ_τ are chosen using the rank estimate on line 22 without adaptation. This recursive construction starts from the leaf level of $\mathcal{T}_S$ and ends at center level l_c.

4.4. Computation of $B_{\tau,\nu}$ at level l_c. It follows from (3.13) that for each node τ at level l_c of $\mathcal{T}_T$ and node ν at level $(L-l_c)$ of $\mathcal{T}_S$, $K(T_\tau, S_\nu)$ is approximated as

$$K(T_\tau, S_\nu) \approx U_{\tau,\nu} B_{\tau,\nu} V_{\tau,\nu}^\top.$$

Using the existing multiplication results $K(T_\tau, S_\nu)\Omega_\nu$, $B_{\tau,\nu}$ can be estimated as

$$\begin{aligned} B_{\tau,\nu} &= \operatorname*{argmin}_{B \in \mathbb{R}^{r_{\tau,\nu} \times r_{\tau,\nu}}} \| \left(K(T_\tau, S_\nu)\Omega_\nu\right) - U_{\tau,\nu} B V_{\tau,\nu}^\top \Omega_\nu \|_F \\ &= U_{t,\nu}^\top \left(K(T_t, S_\nu)\Omega_\nu\right) \left(V_{\tau,\nu}^\top \Omega_\nu\right)^\top. \end{aligned} \tag{4.1}$$

Note that Algorithm 4.1 computes and stores $K(T_\tau, S_\nu)\bar{\Omega}_\nu$ (or $K(T_\tau, S_\nu)^\top \bar{\Gamma}_\tau$) only for one Ω_ν at a time, therefore the algorithm is memory efficient. As an example, Figure 3.3 illustrates the procedure for constructing a 4-level hybrid factorization. Note that the random matrices for the inner factors are structured.

4.5. Cost Analysis. Let $c(n)$ denote the number of operations for the black-box multiplication of $K(S,T)\Omega$ or $K(T,S)^\top\Omega$ for an arbitrary $n \times 1$ vector Ω. In the best-case scenario, $c(n) = \mathcal{O}(n \log n)$ as $K(T,S)$ is typically stored in a compressed form using $\mathcal{O}(n \log n)$ storage units; in the worst-case scenario, $c(n) = \mathcal{O}(n^2)$ when

the matrix is explicitly stored in full. In what follows, we assume $c(n) = \mathcal{O}(n \log n)$. Let $r = \max_{\tau,\nu}\{r_{\tau,\nu}\}$ denote the maximum butterfly rank. Here we analyze the computation and memory costs of Algorithm 4.1 when applied to two classes of butterfly-compressible matrices: (i) r is constant (up to a logarithmic factor). (ii) $r = \mathcal{O}(n^{1/4})$.

4.5.1. r is constant. This case typically occurs when the matrix arises from the discretization of 2D surface integral equations exploiting strong or weak admissibility conditions [7, 20, 21], 3D surface integral equation solvers using strong admissibility [8], low-dimensional Fourier operators [14], etc. For example, Figure 4.1(a) shows the center-level partitioning of a 2D curve used in surface integral-based Helmholtz equation solvers in which all blocks have constant rank except for $\mathcal{O}(1)$ ones with rank $\mathcal{O}(\log n)$. (see [21] for a proof).

As described in subsection 4.1, there are $2^l(r+p)$ black-box matrix-vector multiplications by $K(T,S)^\top$ at level $l = 0, \ldots, l_c$ and $2^{(L-l)}(r+p)$ black-box matrix-vector multiplications by $K(T,S)$ at level $l = L, \ldots, l_c$. Therefore the black-box multiplications require a total of $2(r+p)(1+2+\ldots+2^{l_c})c(n) = \mathcal{O}(rn^{1/2}c(n)) = \mathcal{O}(rn^{3/2}\log n)$ operations. It is worth noting that the multiplications on lines 26 and 37 only involve partial factors V_{p_τ,ν_a} and U_{τ_a,p_ν} and their computational cost is dominated by that of the black-box multiplications. In addition, the algorithm only stores multiplication results for each random matrix of dimensions $n \times (r+p)$ and the computed butterfly factors. The computation and memory costs of Algorithm 4.1 therefore scale as $\mathcal{O}(n^{3/2}\log n)$ and $\mathcal{O}(n \log n)$, respectively.

4.5.2. r is $\mathcal{O}(n^{1/4})$. This case often results from discretizing 3D surface integral equations using weak admissibility. For example, Figure 4.1(b) shows the center-level partitioning of a 3D surface used in surface integral methods for Helmholtz equations. Out of the $16 \times 16 = 256$ center-level blocks of size $\mathcal{O}(n^{1/2}) \times \mathcal{O}(n^{1/2})$, only 4 blocks have rank $\mathcal{O}(n^{1/4})$ representing interactions between adjacent pairs. As the adjacent pair is typically co-planar, the edge shared by the pair can serve as the proxy surface that represent interactions between the pair. Therefore the rank is bounded by the edge dimension $\mathcal{O}(n^{1/4})$ (See [4]). We claim, without proof, that except for $\mathcal{O}(n^{1/4})$ blocks with rank of at most $\mathcal{O}(n^{1/4})$ at each level, the rest has constant rank.

We only need to focus on the construction of blocks with non-constant ranks as the rest has been analyzed above. We first analyze the cost of black-box multiplications on lines 4 and 23. For all blocks at level $0 \le l \le l_c$, denote $N_l = \{\tau | r_{\tau,\nu}\ \textit{non constant}\}$. It can be verified that $|N_l| = \mathcal{O}(2^{\lfloor l/2 \rfloor})$ and each $\tau \in N_l$ requires $\mathcal{O}(r+p) = \mathcal{O}(n^{1/4})$ matrix-vector multiplications. Therefore the black-box multiplications $K(T_\tau, S)^\top \Gamma_\tau$ require a total of $\sum_{l=0}^{l_c} |N_l| \mathcal{O}(n^{1/4}) \mathcal{O}(n \log n) = \mathcal{O}(n^{1/2}c(n)) = \mathcal{O}(n^{3/2}\log n)$ operations. A similar number of operations is required for $K(T, S_\nu)\Omega_\nu$ on line 13 and 33. In addition, since each computed factor has $\mathcal{O}(n^{1/4})$ blocks of rank $\mathcal{O}(n^{1/4})$, the storage cost is $\mathcal{O}(n^{1/4})\mathcal{O}(n^{1/2})L \le \mathcal{O}(n \log n)$. Overall the computation and memory costs of Algorithm 4.1 are of the same orders as those in the case of constant r.

5. Error analysis. In Algorithm 4.1, the low-rank approximation of each individual block constructed by Algorithm 2.1 has $O(\varepsilon)$ relative error with high probability, where ε is the specified relative error tolerance. As a result, the relative error of the constructed butterfly factorization is also probabilistic. Rigorously analyzing such a probabilistic error is out of the scope of this paper. To partially analyze the proposed algorithm, we instead provide an error analysis for a deterministic variant (with minor modifications) of Algorithm 4.1. Specifically, we replace the random matrices $\Gamma_t \in \mathbb{R}^{|T_\tau| \times (r+p)}$, $\Omega_s \in \mathbb{R}^{|S_\nu| \times (r+p)}$ with identity matrices $I \in \mathbb{R}^{|T_\tau| \times |T_\tau|}$,

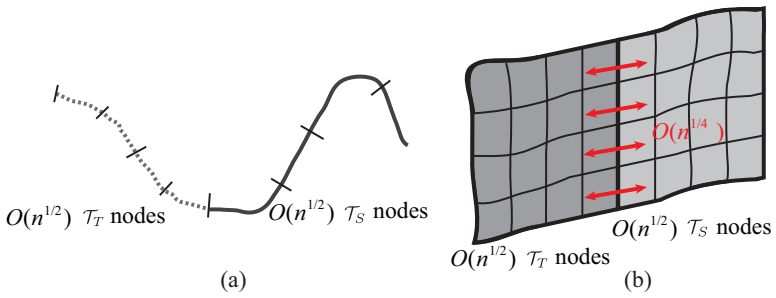

FIG. 4.1. *(a) Center-level geometrical partitioning of a 2D curve for a 4-level butterfly factorization of a weak-admissible matrix. The ranks for all center-level blocks are bounded by $\mathcal{O}(\log n)$. (b) Center-level geometrical partitioning of a 3D surface for a 8-level butterfly factorization of a weak-admissible matrix. There are $\mathcal{O}(n^{1/4})$ center-level blocks with rank $\mathcal{O}(n^{1/4})$ (with the interaction pairs denoted by the red arrows)*

$I \in \mathbb{R}^{|S_\nu| \times |S_\nu|}$ in Algorithm 4.1. On the one hand, this deterministic variant needs to multiply many more vectors than necessary (note that this variant is only used for error analysis purpose). On the other hand, black-box multiplication with identity explicitly forms each block A and Algorithm 2.1 reduces to a deterministic QR decomposition algorithm. The resulting approximation of A thus has a deterministic error bound.

Denote ϵ_0 as the maximum relative error $\|A - UU^\top A\|_F / \|A\|_F$ of all the blocks A in Algorithm 4.1 that are directly compressed by Algorithm 2.1. Given the relative error tolerance ϵ, in the above deterministic variant, $\epsilon_0 \sim \mathcal{O}(\epsilon)$; and in the randomized Algorithm 4.1, ϵ_0 can be viewed as a random variable and is of scale $\mathcal{O}(\epsilon)$ only with probability. In the following, we provide an error analysis using the deterministic variant of Algorithm 4.1.

Recall that the construction of all three forms of butterfly factorizations starts from the leaf level of either $\mathcal{T}_T$ or $\mathcal{T}_S$ and moves towards to the root level. In this procedure, the corresponding low-rank blocks at each level are compressed level-by-level by the nested approach. The error analysis below proceeds in the same manner as we first analyze the error induced by the compression of blocks at the leaf level and we then accumulate the error induced at each non-leaf level with those from the previous levels.

5.1. Approximation error for the nested basis $U_{\tau,\nu}$. Given any level $l_c \leq l \leq L$ of $\mathcal{T}_T$, matrix $K(T,S)$ is partitioned into blocks $\{K(T_\tau, S_\nu)\}$ with τ at level l of $\mathcal{T}_T$ and ν at level $(L-l)$ of $\mathcal{T}_S$, and $K(T_\tau, S_\nu)$ is approximated by $U_{\tau,\nu}U_{\tau,\nu}^\top K(T_\tau, S_\nu)$ where $U_{\tau,\nu}$ is the nested basis computed in subsection 4.2. Theorem 5.1 states that this approximation for all blocks at level l has an error bounded by $\epsilon_0\sqrt{L-l+1}\|A\|_F$.

THEOREM 5.1. *Given a level l in the range $L \geqslant l \geqslant l_c$, the low-rank approximations of all the blocks $K(T_\tau, S_\nu)$ at level l (τ at level l of $\mathcal{T}_T$ and ν at level $(L-l)$ of $\mathcal{T}_S$) based on the computed column basis matrices $U_{\tau,\nu}$ have approximation errors bounded by*

$$\sum_{\tau,\nu \text{ at level } l} \|K(T_\tau, S_\nu) - U_{\tau,\nu}U_{\tau,\nu}^\top K(T_\tau, S_\nu)\|_F^2 \leqslant (L-l+1)\epsilon_0^2 \|K(T,S)\|_F^2, \tag{5.1}$$

where the summation of τ is over the nodes at level l of $\mathcal{T}_T$ and the summation of ν is over the nodes at level $(L-l)$ of $\mathcal{T}_S$.

Proof. We prove the theorem by induction for level l changing from L to l_c. First, for $l = L$, the column basis matrix $U_{\tau,s}$ for block $K(T_\tau, S_s)$ at level L of $\mathcal{T}_T$ is directly computed by Algorithm 2.1. Thus, based on the above relative error assumption, the approximation error is bounded by

$$\|K(T_\tau, S_s) - U_{\tau,s}U_{\tau,s}^\top K(T_\tau, S_s)\|_F^2 \leqslant \epsilon_0^2 \|K(T_\tau, S_s)\|_F^2.$$

Moreover, summing over all nodes τ on both sides of the above inequality proves (5.1) for $l = L$.

Assume that (5.1) holds true for level $(l+1)$. For each node τ at level l of $\mathcal{T}_T$, let ν_1 and ν_2 be siblings at level $(L-l)$ of $\mathcal{T}_S$. Let $\{\tau_1, \tau_2\}$ be the children of τ at level $(l+1)$ of $\mathcal{T}_S$ and ν be the parent node of $\{\nu_1, \nu_2\}$ at level $(L-l-1)$ of $\mathcal{T}_S$. Note that $K(T_\tau, S_{\nu_1})$ and $K(T_\tau, S_{\nu_2})$ are compressed blocks at level l while $K(T_{\tau_1}, S_\nu)$ and $K(T_{\tau_2}, S_\nu)$ are compressed blocks at level $l+1$. Moreover, these two sets of blocks correspond to the same large block in $K(T, S)$, i.e.,

$$\begin{bmatrix} K(T_\tau, S_{\nu_1}) & K(T_\tau, S_{\nu_2}) \end{bmatrix} = \begin{bmatrix} K(T_{\tau_1}, S_\nu) \\ K(T_{\tau_2}, S_\nu) \end{bmatrix} = K(T_\tau, S_\nu).$$

Recall that U_{τ,ν_1} is computed in a nested fashion as

$$\begin{aligned} K(T_\tau, S_{\nu_1}) &= \begin{bmatrix} K(T_{\tau_1}, S_{\nu_1}) \\ K(T_{\tau_2}, S_{\nu_1}) \end{bmatrix} \approx \begin{bmatrix} U_{\tau_1,\nu} & \\ & U_{\tau_2,\nu} \end{bmatrix} \begin{bmatrix} U_{\tau_1,\nu}^\top K(T_{\tau_1}, S_{\nu_1}) \\ U_{\tau_2,\nu}^\top K(T_{\tau_2}, S_{\nu_1}) \end{bmatrix} \\ &\approx \begin{bmatrix} U_{\tau_1,\nu} & \\ & U_{\tau_2,\nu} \end{bmatrix} R_{\tau,\nu_1} R_{\tau,\nu_1}^\top \begin{bmatrix} U_{\tau_1,\nu}^\top K(T_{\tau_1}, S_{\nu_1}) \\ U_{\tau_2,\nu}^\top K(T_{\tau_2}, S_{\nu_1}) \end{bmatrix} = U_{\tau,\nu_1} U_{\tau,\nu_1}^\top K(T_\tau, S_{\nu_1}), \end{aligned}$$

where R_{τ,ν_1} is computed by applying Algorithm 2.1 to the intermediate matrix above, i.e.,

$$\begin{bmatrix} U_{\tau_1,\nu}^\top K(T_{\tau_1}, S_{\nu_1}) \\ U_{\tau_2,\nu}^\top K(T_{\tau_2}, S_{\nu_1}) \end{bmatrix} \approx R_{\tau,\nu_1} R_{\tau,\nu_1}^\top \begin{bmatrix} U_{\tau_1,\nu}^\top K(T_{\tau_1}, S_{\nu_1}) \\ U_{\tau_2,\nu}^\top K(T_{\tau_2}, S_{\nu_1}) \end{bmatrix}. \tag{5.2}$$

The approximation error of block $K(T_\tau, S_{\nu_1})$ at level l can be estimated as

$$\begin{aligned} &\|K(T_\tau, S_{\nu_1}) - U_{\tau,\nu_1}U_{\tau,\nu_1}^\top K(T_\tau, S_{\nu_1})\|_F^2 \\ &= \left\| K(T_\tau, S_{\nu_1}) - \begin{bmatrix} U_{\tau_1,\nu} & \\ & U_{\tau_2,\nu} \end{bmatrix} \begin{bmatrix} U_{\tau_1,\nu}^\top K(T_{\tau_1}, S_{\nu_1}) \\ U_{\tau_2,\nu}^\top K(T_{\tau_2}, S_{\nu_1}) \end{bmatrix} \right\|_F^2 \\ &\quad + \left\| \begin{bmatrix} U_{\tau_1,\nu} & \\ & U_{\tau_2,\nu} \end{bmatrix} \begin{bmatrix} U_{\tau_1,\nu}^\top K(T_{\tau_1}, S_{\nu_1}) \\ U_{\tau_2,\nu}^\top K(T_{\tau_2}, S_{\nu_1}) \end{bmatrix} - U_{\tau,\nu_1}U_{\tau,\nu_1}^\top K(T_\tau, S_{\nu_1}) \right\|_F^2 \\ &= \left\|K(T_{\tau_1}, S_{\nu_1}) - U_{\tau_1,\nu}U_{\tau_1,\nu}^\top K(T_{\tau_1}, S_{\nu_1})\right\|_F^2 + \left\|K(T_{\tau_2}, S_{\nu_1}) - U_{\tau_2,\nu}U_{\tau_2,\nu}^\top K(T_{\tau_2}, S_{\nu_1})\right\|_F^2 \\ &\quad + \left\| \begin{bmatrix} U_{\tau_1,\nu}^\top K(T_{\tau_1}, S_{\nu_1}) \\ U_{\tau_2,\nu}^\top K(T_{\tau_2}, S_{\nu_1}) \end{bmatrix} - R_{\tau,\nu_1} R_{\tau,\nu_1}^\top \begin{bmatrix} U_{\tau_1,\nu}^\top K(T_{\tau_1}, S_{\nu_1}) \\ U_{\tau_2,\nu}^\top K(T_{\tau_2}, S_{\nu_1}) \end{bmatrix} \right\|_F^2 \end{aligned}$$

where the first equation is due to the orthogonality between the columns from the two bracketed terms, and the second equation follows from the unitary invariance of the Frobenius norm, as $U_{\tau_1,\nu}$ and $U_{\tau_2,\nu}$ both have orthonormal columns. The last term above, which is the approximation error of (5.2), is guaranteed to satisfy

$$\epsilon_0^2 \left\| \begin{bmatrix} U_{\tau_1,\nu}^\top K(T_{\tau_1}, S_{\nu_1}) \\ U_{\tau_2,\nu}^\top K(T_{\tau_2}, S_{\nu_1}) \end{bmatrix} \right\|_F^2 \leqslant \epsilon_0^2 \left\| \begin{bmatrix} K(T_{\tau_1}, S_{\nu_1}) \\ K(T_{\tau_2}, S_{\nu_1}) \end{bmatrix} \right\|_F^2 = \epsilon_0^2 \|K(T_\tau, S_{\nu_1})\|_F^2.$$

Similarly, we can estimate the approximation error of block $K(T_\tau, S_{\nu_2})$ at level l as

$$\begin{aligned}
&\|K(T_\tau, S_{\nu_2}) - U_{\tau,\nu_2} U_{\tau,\nu_2}^\top K(T_\tau, S_{\nu_2})\|_F^2 \\
&\leqslant \left\|K(T_{\tau_1}, S_{\nu_2}) - U_{\tau_1,\nu} U_{\tau_1,\nu}^\top K(T_{\tau_1}, S_{\nu_2})\right\|_F^2 \\
&\qquad + \left\|K(T_{\tau_2}, S_{\nu_2}) - U_{\tau_2,\nu} U_{\tau_2,\nu}^\top K(T_{\tau_2}, S_{\nu_2})\right\|_F^2 + \epsilon_0^2 \|K(T_\tau, S_{\nu_2})\|_F^2.
\end{aligned}$$

Assembling the above three inequalities and using $S_\nu = S_{\nu_1} \cup S_{\nu_2}$, we obtain

$$\begin{aligned}
&\|K(T_\tau, S_{\nu_1}) - U_{\tau,\nu_1} U_{\tau,\nu_1}^\top K(T_\tau, S_{\nu_1})\|_F^2 + \|K(T_\tau, S_{\nu_2}) - U_{\tau,\nu_2} U_{\tau,\nu_2}^\top K(T_\tau, S_{\nu_2})\|_F^2 \\
&\leqslant \left\|K(T_{\tau_1}, S_\nu) - U_{\tau_1,\nu} U_{\tau_1,\nu}^\top K(T_{\tau_1}, S_\nu)\right\|_F^2 + \left\|K(T_{\tau_2}, S_\nu) - U_{\tau_2,\nu} U_{\tau_2,\nu}^\top K(T_{\tau_2}, S_\nu)\right\|_F^2 \\
&\qquad + \epsilon_0 \left(\|K(T_\tau, S_{\nu_1})\|_F^2 + \|K(T_\tau, S_{\nu_2})\|_F^2\right).
\end{aligned}$$

The left hand side of the inequality is the approximation error of the two blocks at level l while the first two terms on the right hand side are the approximation errors of two blocks at level $(l+1)$. Summing over all the nodes τ at level l of $\mathcal{T}_T$ and all the node pairs (ν_1, ν_2) at level $(L-l)$ of $\mathcal{T}_S$ on both sides of this inequality, we obtain,

$$\begin{aligned}
&\sum_{\tau,\nu \text{ at level } l} \|K(T_\tau, S_\nu) - U_{\tau,\nu} U_{\tau,\nu}^\top K(T_\tau, S_\nu)\|_F^2 \\
&\leqslant \sum_{\tau,\nu \text{ at level } (l+1)} \|K(T_\tau, S_\nu) - U_{\tau,\nu} U_{\tau,\nu}^\top K(T_\tau, S_\nu)\|_F^2 + \epsilon_0^2 \|K(T, S)\|_F^2 \\
&\leqslant (L - (l+1) + 1)\epsilon_0^2 \|K(T, S)\|_F^2 + \epsilon_0^2 \|K(T, S)\|_F^2 = (L - l + 1)\epsilon_0^2 \|K(T, S)\|_F^2. \square
\end{aligned}$$

5.2. Approximation error for the nested basis $V_{\tau,\nu}$. The overall approximation error for the projection to the row bases is bounded just like that of the column bases shown in Theorem 5.2.

THEOREM 5.2. *Given a level l in the range $L \geqslant l \geqslant l_c$, the low-rank approximations of all the blocks $K(T_\tau, S_\nu)$ at level l (τ at level l of $\mathcal{T}_T$ and ν at level $(L-l)$ of $\mathcal{T}_S$) based on the computed row basis matrices $V_{\tau,\nu}$ have approximation errors bounded as*

$$\sum_{\tau,\nu \text{ at level } l} \|K(T_\tau, S_\nu) - K(T_\tau, S_\nu) V_{\tau,\nu} V_{\tau,\nu}^\top\|_F^2 \leqslant (l+1)\epsilon_0^2 \|K(T, S)\|_F^2, \tag{5.3}$$

where the summation of τ is over the nodes at level l of $\mathcal{T}_T$ and the summation of ν is over the nodes at level $(L-l)$ of $\mathcal{T}_S$.

Proof. The proof is similar to that of Theorem 5.1. ∎

5.3. Approximation error for the overall factorization. Combining the above two error analyses for the column- and row-wise butterfly factorizations, the overall approximation error of a hybrid butterfly factorization at any level l can be bounded as shown in Theorem 5.3

THEOREM 5.3. *Given center level $l_c = \lfloor L/2 \rfloor$, the low-rank approximations of all the blocks $K(T_\tau, S_\nu)$ at level l_c (τ at level l_c of $\mathcal{T}_T$ and ν at level $(L-l_c)$ of $\mathcal{T}_S$) based on the computed basis matrices $U_{\tau,\nu}$ and $V_{\tau,\nu}$ have an overall approximation error bounded by*

$$\sum_{\tau,\nu \text{ at level } l} \|K(T_\tau, S_\nu) - U_{\tau,\nu} U_{\tau,\nu}^\top K(T_\tau, S_\nu) V_{\tau,\nu} V_{\tau,\nu}^\top\|_F^2 \leqslant (L+2)\epsilon_0^2 \|K(T, S)\|_F^2, \tag{5.4}$$

where the summation of τ is over the nodes at level l_c of $\mathcal{T}_T$ and the summation of ν is over the nodes at level $(L - l_c)$ of $\mathcal{T}_S$.

Proof. Every block $K(T_\tau, S_\nu)$ has its approximation error bounded as

$$\begin{aligned}
&\|K(T_\tau, S_\nu) - U_{\tau,\nu}U_{\tau,\nu}^\top K(T_\tau, S_\nu)V_{\tau,\nu}V_{\tau,\nu}^\top\|_F^2 \\
&\leqslant \|K(T_\tau, S_\nu) - U_{\tau,\nu}U_{\tau,\nu}^\top K(T_\tau, S_\nu)\|_F^2 \\
&\qquad + \|U_{\tau,\nu}U_{\tau,\nu}^\top K(T_\tau, S_\nu) - U_{\tau,\nu}U_{\tau,\nu}^\top K(T_\tau, S_\nu)V_{\tau,\nu}V_{\tau,\nu}^\top\|_F^2 \\
&\leqslant \|K(T_\tau, S_\nu) - U_{\tau,\nu}U_{\tau,\nu}^\top K(T_\tau, S_\nu)\|_F^2 + \|K(T_\tau, S_\nu) - K(T_\tau, S_\nu)V_{\tau,\nu}V_{\tau,\nu}^\top\|_F^2.
\end{aligned}$$

Using (5.1) and (5.3) in the above equation, inequality (5.4) is proven. ∎

The above error analysis can be further combined with the probability of $\epsilon_0 \sim O(\epsilon)$ in the randomized Algorithm 4.1 to derive a rigorous probability bound. Suppose Algorithm 2.1 can attain a relative error $\mathcal{O}(\epsilon)$ with probability ρ, one can show that Algorithm 4.1 can attain the relative error in Theorem 5.3 (replacing ϵ_0 with $\mathcal{O}(\epsilon)$) with probability at least $\rho^{2^L(L+2)}$. Though we will not derive a tighter probability bound in this paper, the numerical results in section 7 provide clear evidence for the accuracy of the randomized butterfly factorization algorithm.

Lastly, we note that the above error analysis also applies to other deterministic projection (i.e., SVD and QR)-based butterfly factorization algorithm. The only assumption we made in Theorem 5.3 is that the low-rank approximation of all the blocks explicitly compressed in the construction of a butterfly factorization has the actual relative error bounded by ϵ_0.

6. Parallelization. This section outlines a distributed-memory implementation of Algorithm 4.1. We first consider the task of parallelizing a butterfly-vector multiplication assuming that the butterfly factors are stored in a distributed fashion. Note that this is the dominant computational task in the proposed algorithm as the black-box multiplications on lines 4, 23, 13, 33 often involve existing butterfly representations, and the explicit multiplications on lines 34, 26, 37 involve partial butterfly factors. A good parallelization strategy for these two types of multiplications therefore is paramount to the parallel implementation of the proposed randomized butterfly reconstruction scheme.

Without loss of generality, we assume that $r_{\tau,\nu} = r$ for some constant r, the number of butterfly levels L is even, $n = r2^L$, and the number of processes $1 \leq p \leq 2^L$ is a power of two. To estimate the algorithm's communication cost, we only analyze the number of messages and communication volume as we assume the time to communicate a message of size m between two processes is $\alpha + \beta m$ where α and β represent message latency and inverse bandwidth, respectively [11].

6.1. Column/Row-wise factorization. We first describe the parallelization scheme for the column-wise butterfly factorization studied in [29]. The parallel data layout can be descried as follows: (i) starting from level L of $\mathcal{T}_T$, one process stores $2^L/p$ consecutive blocks $U_{\tau,s}$ following a 1D-row layout. (ii) At levels $l = L-1, \ldots, 1$, let $\{\tau_1, \tau_2\}$ be the children of τ at level l of $\mathcal{T}_T$ and $\{\nu_1, \nu_2\}$ be the children of ν at level $L - l - 1$ of $\mathcal{T}_S$. Consider the *combined* transfer matrix:

$$R_{\tau,\nu} = \begin{bmatrix} R_{\tau_1,\nu} & \\ & R_{\tau_2,\nu} \end{bmatrix} \begin{bmatrix} R_{\tau,\nu_1} & R_{\tau,\nu_1} \end{bmatrix} \tag{6.1}$$

where $R_{\tau_a,\nu}$ can be replaced by $U_{\tau_a,\nu}$ if $l = L - 1$. The parallelization scheme stores $R_{\tau_a,\nu}$ (or $U_{\tau_a,\nu}$ if $l = L - 1$) and R_{τ,ν_a} on the same process. (iii) At level $l = 0$, the

blocks $E_{t,\nu}$ and $R_{t,\nu}$ are stored on the same process. Figure 6.1(a) illustrates the data layout for a 4-level column-wise butterfly factorization using 4 processes. Note that at level $l = 0$, the layout of $E_{t,\nu}$ is similar to the 1D-row layout for $U_{\tau,s}$, but in an index-reversed order.

When multiplying a (partial) butterfly stored as described above by a vector stored using the 1D-row layout, an all-to-all communication is required to convert the vector from the 1D-row layout to the index-reversed layout of $E_{t,\nu}$. To this end, each process communicates $\min\{2^L/p, p-1\}$ messages of total size $r2^L/p$. For the example in Figure 6.1(a), process 0 (light green) needs to scatter the locally stored part of Ω of size $2^L/p \times r$ to all other processes before the multiplication operation with E^0 can take place. After that, there are $\log p$ levels requiring pair-wise exchanges of the intermediate multiplication results $R^l \dots R^0 E^0 \Omega$ (e.g., after multiplication with R_0 and R_1 in Figure 6.1(a)). For each exchange operation, reductions involving messages of size $r2^L/p$ between two processes are performed (note: broadcast is conducted for multiplying the transpose $K(T,S)^\top$). For example, considering the multiplication with R_{τ,ν_1} (stored on process 0 in light green) and R_{τ,ν_2} (stored on process 2 in dark green) in the first diagonal block of R^0, the local multiplication results of size $2r \times r$ require a reduction between processes 0 and 2 before the local multiplication with the first diagonal block of R^1 is performed. The communication costs are listed in Table 6.1.

The parallelization of the row-wise butterfly factorization can be described similarly: At level $l = 0$ of $\mathcal{T}_0$, $V_{t,\nu}$ is stored using the 1D-row layout; at levels $l = 1, \dots, L$, W_{τ,ν_a} (or V_{τ,ν_a} if $l = 1$) and $W_{\tau_a,\nu}$ are stored on the same process. The communication costs for the row-wise parallelization are the same as those for the column-wise parallelization.

6.2. Hybrid factorization. Since Algorithm 4.1 computes a hybrid factorization, it is more convenient and efficient to combine the column-wise and row-wise parallel data layouts. Specifically, the factors $U^L R^{L-1} R^{L-2} \dots R^{l_c}$ are stored using the column-wise layout whereas those of $W^{l_c} W^{l_c-1} \dots W^1 V^0$ adhere to the row-wise layout. The block $B_{\tau,\nu}$ is handled by the same process as $W_{\tau,\nu}$ in W^{l_c}. When multiplying the (partial) butterfly with a vector Ω, all-to-all communication is needed for the intermediate result $B^{l_c} W^{l_c} W^{l_c-1} \dots W^1 V^0 \Omega$. In contrast to the column/row-wise layout, the number of levels requiring pair-wise exchange is only $\max\{0, \log p^2/2^L\}$. Note that no exchange is needed if $p \leq 2^{L/2}$. As an example, Figure 6.1(b) shows the multiplication of a 4-level hybrid butterfly using 4 processes. Here, $p = 2^{L/2}$ and the only communication is the all-to-all operation that occurs after multiplying with B^2 to switch from the row- to the column-wise layout. Table 6.1 lists the communication costs for the hybrid data layout. Clearly, the multiplication using the hybrid factorization requires less communication than multiplication with the column/row-wise factorization.

Now we can summarize the proposed parallelization strategy for Algorithm 4.1 as follows:

- The black-box multiplications on lines 4, 23, 13, 33 follow the hybrid layout if they involve existing parallel butterfly representations.
- The computed butterfly factors follow the hybrid layout for each process that applies Algorithm 2.1 on lines 26 and 37 locally for the blocks it is in charge of.
- The explicit multiplications with computed butterfly blocks on lines 34, 26, 37 follow the hybrid layout but may not involve all the processes.

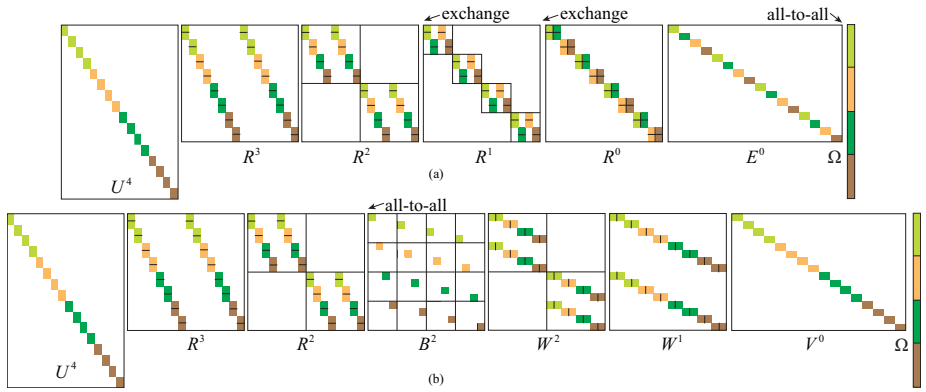

FIG. 6.1. *Parallel data layout for distributing a 4-level (a) column-wise and (b) hybrid factorization with 4 processes. Each color represents one process. The arrows denote places where exchange and all-to-all communications are needed for multiplying the butterfly with a 1D row distributed vector. Note that no exchange is needed for the hybrid butterfly.*

	exchange		all-to-all	
	volume	message count	volume	message count
column/row	$\frac{r2^L \log p}{p}$	$\log p$	$\frac{r2^L}{p}$	$\min\{\frac{2^L}{p}, p-1\}$
hybrid	$\frac{r2^L}{p} \max\{0, \log \frac{p^2}{2^L}\}$	$\max\{0, \log \frac{p^2}{2^L}\}$	$\frac{r2^L}{p}$	$\min\{\frac{2^L}{p}, p-1\}$

TABLE 6.1
Communication volume and message counts for one matrix-vector multiplication.

7. Numerical results. This section provides several examples to demonstrate the accuracy and efficiency of the proposed randomized algorithm. The accuracy of the proposed algorithms is characterized by

$$error = \frac{\left\| A\Omega - \left(U^L R^{L-1} R^{L-2} \dots R^l\right) B^l \left(W^l W^{l-1} \dots W^1 V^0\right) \Omega \right\|_F}{\|A\Omega\|_F} \tag{7.1}$$

with a random testing matrix Ω of 16 columns. All experiments are performed on the Cori Haswell machine at NERSC, which is a Cray XC40 system and consists of 2388 dual-socket nodes with Intel Xeon E5-2698v3 processors running 16 cores per socket. The nodes are configured with 128 GB of DDR4 memory clocked at 2133 MHz. Unless stated otherwise, all experiments use one Cori node.

7.1. Exact butterfly factorization. We first apply the algorithm to "recover" a matrix with known butterfly factorization. In what follows, we use the symbol "$\bar{a}$" to differentiate the blocks and factors of the known butterfly from the computed ones. Let A be a $n \times n$ matrix with a given L-level butterfly factorization $A = (\bar{U}^L \bar{R}^{L-1} \bar{R}^{L-2} \dots \bar{R}^l) \bar{B}^l (\bar{W}^l \bar{W}^{l-1} \dots \bar{W}^1 \bar{V}^0)$ that has $r_{\tau,\nu} = r$ for all blocks at all levels for some constant r. The blocks $\bar{W}_{\tau,\nu}$, $\bar{R}_{\tau,\nu}$, and $\bar{B}_{\tau,\nu}$ have dimensions $r \times 2r$, $2r \times r$ and $r \times r$, respectively. We choose the matrix dimension $n = 2^{L+3}$ such that $\bar{U}_{\tau,s}$ and $\bar{V}_{t,\nu}$ have dimensions $8 \times r$. The blocks $\bar{U}_{\tau,s}$, $\bar{V}_{t,\nu}$, $\bar{W}_{\tau,\nu}$, $\bar{R}_{\tau,\nu}$ are constructed as random unitary matrices, and the entries of $\bar{B}_{\tau,\nu}$ are random variables which are independent and identically distributed, following a normal distribution. We use this explicit representation to perform "black-box" matrix-vector multiplications and apply Algorithm 4.1 to retrieve the butterfly factorization.

The memory costs for varying n with fixed $r = 8$ when using the proposed algorithm and the randomized SVD-based reference algorithm in [14] are plotted in Figure 7.1(a). We set $p = 2$ and $r_0 = 4$ in Algorithm 4.1. The proposed algorithm requires only $\mathcal{O}(n\log n)$ memory as opposed to the reference algorithm, which requires $\mathcal{O}(n^{1.5})$ memory.

Next, the performance of the proposed parallelization scheme is demonstrated by applying Algorithm 4.1 to the butterfly representation with $n = 2.56 \times 10^6$ and $r = 10$. Figure 7.1(b) plots the runtimes of Algorithm 4.1 and a single matvec with the exact butterfly in both hybrid and column-wise data layouts, for process counts ranging from 16 to 2048. As predicted by Table 6.1, the hybrid data layout yields a substantially lower communication cost in both matvec and factorization. As a result, we obtain up to 4x speedups for a single matvec comparing the hybrid layout to the column-wise layout; also, the runtime of Algorithm 4.1 is 25% faster.

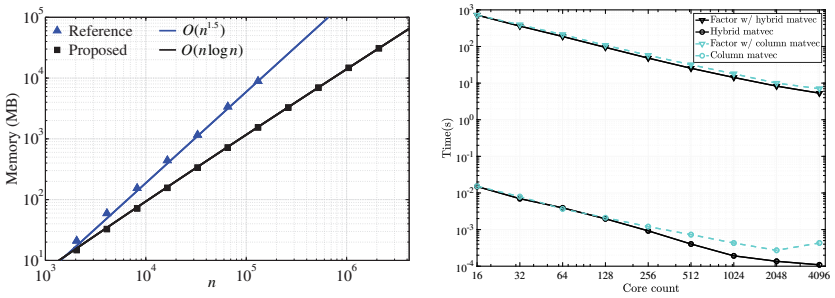

FIG. 7.1. *(a) Memory for applying Algorithm 4.1 and the reference algorithms to an exact butterfly representation. (b) Runtime for applying Algorithm 4.1 and matrix-vector multiplications to the exact butterfly representation with hybrid and column-wise patterns for varying core counts.*

7.2. 2D Helmholtz kernel. Next, consider the following wave scattering example. Let C^1 and C^2 denote two disjoint curves. Suppose C^1 and C^2 are partitioned into n and m constant-sized segments C_i^1 and C_j^2, $i = 1, ..., n$, $j = 1, ..., m$. Let k_0 denote the wavenumber, then $m, n = \mathcal{O}(k_0)$. Consider the following $m \times n$ matrix A

$$A = Z^{21}(Z^{11})^{-1} \tag{7.2}$$

$$Z_{i,j}^{11} = \int_{C_j^1} H_0^{(2)}(k_0|\rho_i^1 - \rho|)d\rho,\ i, j = 1, ..., n \tag{7.3}$$

$$Z_{i,j}^{21} = \int_{C_j^1} H_0^{(2)}(k_0|\rho_i^2 - \rho|)d\rho,\ i = 1, ..., m,\ j = 1, ..., n \tag{7.4}$$

Here, $H_0^2(\cdot)$ is the zeroth-order Hankel function of the second kind, ρ is a position vector on C^1, ρ_i^1 and ρ_i^2 denote the center of segment C_i^1 and C_i^2. In principle, the $n \times n$ matrix $(Z^{11})^{-1}$ relates the equivalent source on C^1 to the incident fields on C^1, and the $m \times n$ matrix Z^{21} computes fields observed on C^2 scattered by the source on C^1. The matrix A in (7.2) resembles the *scattering matrix* as it relates the incident fields on C^1 to its scattered fields on C^2 [10, 20]. In what follows, we seek a butterfly-compressed representation of the matrix A.

n	L	ϵ	r	error	Time (sec)	Memory (MB)
20000	9	3E-04	10	3.63E-04	3.61E+01	1.68E+01
80000	11	3E-04	10	4.22E-04	3.27E+02	7.71E+01
320000	13	3E-04	10	4.98E-04	3.27E+03	3.47E+02
1280000	15	3E-04	10	7.38E-04	3.02E+04	1.58E+03
20000	9	3E-05	12	2.45E-05	7.80E+01	3.13E+01
80000	11	3E-05	12	2.39E-05	7.21E+02	1.32E+02
320000	13	3E-05	12	2.37E-05	6.46E+03	5.99E+02
1280000	15	3E-05	12	3.26E-05	5.87E+04	2.64E+03
20000	9	3E-06	14	8.09E-06	8.10E+01	3.71E+01
80000	11	3E-06	14	8.39E-06	7.82E+02	1.71E+02
320000	13	3E-06	14	9.11E-06	7.17E+03	7.92E+02
1280000	15	3E-06	14	9.57E-06	6.17E+04	3.56E+03

TABLE 7.1

Time, memory and measured error for computing a hybrid factorization of (7.2) using the proposed algorithm with varying matrix size n and tolerance ϵ.

In this example, suppose C^1 and C^2 are two parallel lines with length D and their respective distance D, where D is set to 1 m. It is well known that, for elongated structures [22, 26], Z^{11} and its inverse have compressed representations using $\mathcal{H}$-matrix or other low-rank factorization-based hierarchical techniques, requiring at most $\mathcal{O}(n\log n)$ computation and memory resources [3, 6, 23]. In addition, Z^{21} can be compressed by the butterfly factorization requiring $\mathcal{O}(n\log n)$ computation and memory resources [20, 21]. Therefore, A and its transpose can be applied to any vector in $\mathcal{O}(n\log n)$ operations irrespective of wavenumber k_0.

To compute a butterfly-compressed representation of A using the proposed algorithm, the lengths of line segments C_i^1 and C_i^2 are set to approximately 0.05λ with $\lambda = 2\pi/k_0$ denoting the wavelength. The sizes of the leaf-level point sets T_τ and S_ν are set to approximately 39. The matrices $(Z^{11})^{-1}$ and Z^{21} are compressed respectively with the $\mathcal{H}$-matrix and butterfly factorization with a high accuracy (tolerance $\epsilon = 10^{-8}$), respectively. Note that these compressed representations are used instead of A in (7.1) and in black-box multiplications.

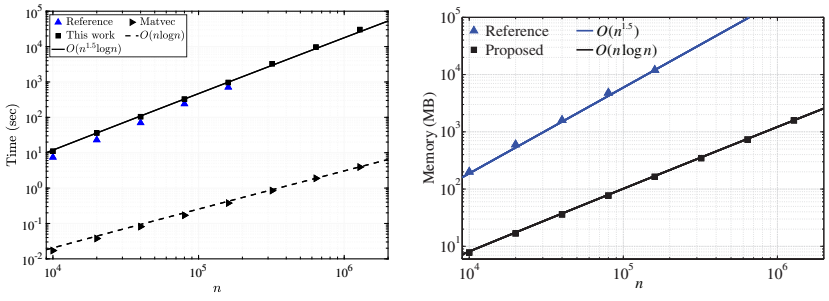

FIG. 7.2. *(a) Factorization and one black-box matvec times and (b) memory for applying Algorithm 4.1 and the reference algorithms to (7.2).*

The computational results with butterfly level $L = 9, 11, 13, 15$ and $\lambda = 9.76 \times 10^{-5}m$, $2.44 \times 10^{-5}m$, $6.10 \times 10^{-6}m$, $1.52 \times 10^{-6}m$ for different tolerances $\epsilon = 3 \times 10^{-4}, 3 \times 10^{-5}, 3 \times 10^{-6}$

are listed in Table 7.1. We set $p = 2$ and $r_0 = 4$ in Algorithm 4.1. The measured error scales proportional to L as predicted in Theorem 5.3. For $\epsilon = 10^{-3}$, the computationally most expensive case (when $n = 1.28 \times 10^6$) requires about 9 h CPU time and 1.6 GB memory. Moreover, the memory cost and the factorization time obey the predicted $\mathcal{O}(n\log n)$ and $\mathcal{O}(n^{1.5}\log n)$ scaling estimates.

The computation time and memory cost with $L = 8, ..., 15$ and $\epsilon = 10^{-3}$ are plotted in Figure 7.2 using the proposed algorithm and the reference algorithm in [14]. The proposed algorithm is slightly slower than the reference algorithm as it requires slightly more testing vectors to reduce the memory cost. That said, the proposed algorithm requires much less memory than the reference algorithm (see Figure 7.2(b)).

7.3. 3D Helmholtz kernel. Finally, we consider the wave interactions between two semi-sphere surfaces C^1 and C^2 of unit radius adjacent to each other. Each semi-sphere is discretized via the Nyström method into n sample points. We seek a butterfly factorization of the following $n \times n$ matrix A:

$$A_{i,j} = \frac{\exp(i2\pi\kappa|\rho_i - \rho_j|)}{|\rho_i - \rho_j|} \tag{7.5}$$

with ρ_i and ρ_j denoting sample point i and j on C^1 and C^2, respectively. The wavenumber κ is set such that $n = 50\kappa^2/\pi$ represents approximately 10 sample points per wavelength. We first compute $A \approx (\bar{U}^L \bar{R}^{L-1} \bar{R}^{L-2} \dots \bar{R}^l)\bar{B}^l(\bar{W}^l \bar{W}^{l-1} \dots \bar{W}^1 \bar{V}^0)$ with $\epsilon = 10^{-11}$ and use the result for (7.1) and the black-box multiplications.

The computational results with butterfly level $L = 8, 10, 12$ for different tolerances $\epsilon = 3 \times 10^{-5}, 3 \times 10^{-7}, 3 \times 10^{-9}, 3 \times 10^{-11}$ are listed in Table 8.1. We set $p = 4$ and $r_0 = 64$ in Algorithm 4.1. These experiments use 2 Cori nodes with a total of 64 MPI processes. The proposed algorithm achieves the desired accuracies predicted by Theorem 5.3. The computation time, memory and observed rank with $\epsilon = 10^{-2}$ are plotted in Figure 8.1 using the proposed algorithm. Despite the $\mathcal{O}(n^{0.25})$ rank scaling (see Figure 4.1(b) as an illustration), Algorithm 4.1 still attains $\mathcal{O}(n^{1.5}\log n)$ computation and $\mathcal{O}(n\log n)$ memory complexities as estimated in subsection 4.5.2. In addition, the estimated memory usage with the reference algorithm is also plotted in Figure 8.1(b), the proposed algorithm requires much less memory in comparison.

7.4. Composition of two Fourier integral operators. Finally, consider the following $n \times n$ matrix which represents discretization of the composition of two Fourier integral operators:

$$A = F^1 K F^2 \tag{7.6}$$

where

$$F^1_{i,j} = \exp(x_i\xi_j + \xi_j \sin(2\pi x_i)/8) \tag{7.7}$$

$$F^2_{i,j} = \exp(x_i\xi_j + x_i^2\xi_j/16) \tag{7.8}$$

represent the two discretized Fourier integral operators with $x_i = (i-1)/n$, $\xi_j = j-1$, $i, j \le n$, and $K_{i,j} = \exp(2\pi(i-1)(j-1)/n)$ is the discretized Fourier transform. We first compress F^1, K, F^2 as three butterflies with $\epsilon = 10^{-6}$ and then use the product of three butterflies for (7.1) and the black-box multiplications.

The computational results with butterfly level $L = 10, 12, 14$ for different tolerances $\epsilon = 3 \times 10^{-3}, 3 \times 10^{-5}, 3 \times 10^{-6}$ are listed in Table 8.2. We set $p = 4$ and

$r_0 = 32$ in Algorithm 4.1. It's worth noting that although F^1, K and F^2 permit butterfly ranks independent of n and L, their product A exhibits increasing butterfly ranks.

The computation time and memory cost with $L = 10, 11, 12, 13, 14$ and $\epsilon = 3 \times 10^{-5}$ are plotted in Figure 8.2 using the proposed algorithm. Despite the rank increase, the proposed algorithms can achieve $\mathcal{O}(n^{1.5}\log n)$ computation and $\mathcal{O}(n\log n)$ memory complexities as estimated in subsection 4.5.2.

8. Conclusion and discussion. This paper presented a fast and memory-efficient randomized algorithm for computing the butterfly factorization of a matrix assuming the availability of a black-box algorithm for applying the matrix and its transpose to a vector. The proposed algorithm applies the matrix and its transpose to structured random vectors to reconstruct the orthonormal row and column bases of judiciously-selected low-rank blocks of the (assumed) butterfly-compressible matrix. The algorithm only requires $\mathcal{O}(n^{1.5}\log n)$ computation and $\mathcal{O}(n\log n)$ storage resources for matrices arising from the integral equation based discretization of both 2D and 3D Helmholtz problems using either weak or strong admissibility separation criteria. The accuracy of the proposed algorithm only weakly depends on the number of butterfly levels. The computation time of the algorithm can be reduced leveraging distributed-memory parallelism. We expect that the proposed algorithm will play an important role in constructing both dense and sparse, fast and parallel, hierarchical matrix-based direct solvers for high-frequency wave equations. The code described here is part of the Fortran/C++ solver package ButterflyPACK (https://github.com/liuyangzhuan/ButterflyPACK), freely available online. The integration of butterfly factorizations into the sparse solver package STRUMPACK is currently in progress.

Acknowledgments. This research was supported in part by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration, and in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program through the FASTMath Institute under Contract No. DE-AC02-05CH11231 at Lawrence Berkeley National Laboratory.

This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231.

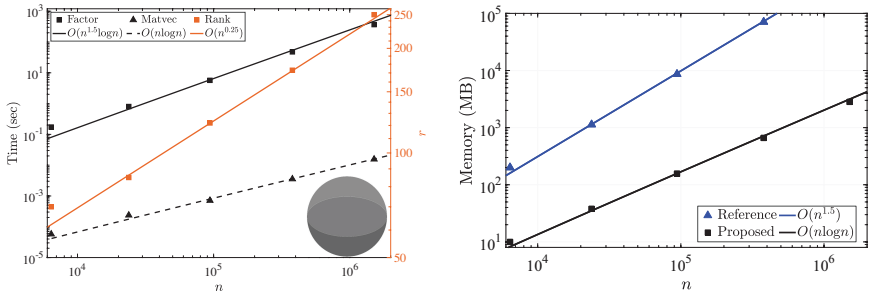

FIG. 8.1. *(a) Times for factorization and one black-box matvec (left y-axis), observed rank (right y-axis) and (b) memory for applying Algorithm 4.1 and the reference algorithms to (7.5).*

n	L	ϵ	r	error	Time (sec)	Memory (MB)
23806	8	3E-05	181	1.14E-04	3.36E+00	1.37E+02
94024	10	3E-05	252	1.29E-04	2.14E+01	5.62E+02
379065	12	3E-05	354	1.52E-04	1.71E+02	2.37E+03
23806	8	3E-07	271	1.09E-06	7.40E+00	3.37E+02
94024	10	3E-07	371	1.22E-06	4.94E+01	1.41E+03
379065	12	3E-07	522	1.31E-06	3.93E+02	5.94E+03
23806	8	3E-09	362	1.05E-08	1.50E+01	6.72E+02
94024	10	3E-09	499	1.11E-08	1.01E+02	2.85E+03
379065	12	3E-09	705	1.25E-08	7.98E+02	1.22E+04
23806	8	3E-11	474	8.46E-11	2.70E+01	1.25E+03
94024	10	3E-11	634	1.69E-10	1.94E+02	5.35E+03
379065	12	3E-11	852	8.74E-11	1.69E+03	2.30E+04

TABLE 8.1

Time, memory and measured error for computing a hybrid factorization of (7.5) using the proposed algorithm with varying matrix size n and tolerance ϵ.

n	L	ϵ	r	error	Time (sec)	Memory (MB)
4000	10	3E-03	26	1.79E-02	6.37E+00	2.24E+01
16000	12	3E-03	43	2.67E-02	6.99E+01	1.13E+02
64000	14	3E-03	89	3.86E-02	9.06E+02	5.82E+02
4000	10	3E-05	31	1.83E-04	1.00E+01	4.02E+01
16000	12	3E-05	54	4.75E-04	1.12E+02	2.04E+02
64000	14	3E-05	115	1.33E-03	1.52E+03	1.01E+03
4000	10	3E-06	29	1.64E-05	1.21E+01	4.72E+01
16000	12	3E-06	61	3.80E-05	1.37E+02	2.50E+02
64000	14	3E-06	123	8.34E-05	1.85E+03	1.29E+03

TABLE 8.2

Time, memory and measured error for computing a hybrid factorization of (7.6) using the proposed algorithm with varying matrix size n and tolerance ϵ.

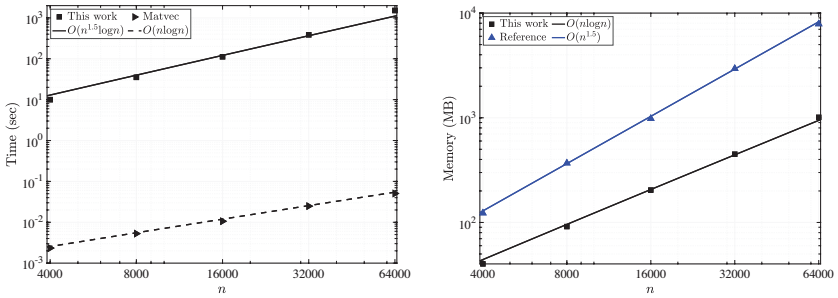

FIG. 8.2. *(a) Times for factorization and one black-box matvec and (b) memory for applying Algorithm 4.1 and the reference algorithms to (7.6).*

REFERENCES

[1] J. BREMER, Z. CHEN, AND H. YANG, *Rapid application of the spherical harmonic transform*

via interpolative decomposition butterfly factorization, 2020, https://arxiv.org/abs/2004.11346.

[2] E. Candès, L. Demanet, and L. Ying, *A fast butterfly algorithm for the computation of Fourier integral operators*, Multiscale Modeling & Simulation, 7 (2009), pp. 1727–1750.

[3] S. Chandrasekaran, P. Dewilde, M. Gu, W. Lyons, and T. Pals, *A fast solver for hss representations via sparse matrices*, SIAM Journal on Matrix Analysis and Applications, 29 (2007), pp. 67–81.

[4] B. Engquist and H. Zhao, *Approximate separability of the green's function of the helmholtz equation in the high frequency limit*, Communications on Pure and Applied Mathematics, 71 (2018), pp. 2220–2274, https://doi.org/10.1002/cpa.21755.

[5] C. Gorman, G. Chávez, P. Ghysels, T. Mary, F.-H. Rouet, and X. S. Li, *Robust and accurate stopping criteria for adaptive randomized sampling in matrix-free hierarchically semiseparable construction*, SIAM Journal on Scientific Computing, 41 (2019), pp. S61–S85.

[6] L. Grasedyck and W. Hackbusch, *Construction and arithmetics of H-matrices*, Computing, 70 (2003), pp. 295–334.

[7] H. Guo, J. Hu, and E. Michielssen, *On MLMDA/butterfly compressibility of inverse integral operators*, IEEE Antennas and Wireless Propagation Letters, 12 (2013), pp. 31–34.

[8] 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 (2017), pp. 4742–4750.

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

[10] S. Hao, P. Martinsson, and P. Young, *An efficient and highly accurate solver for multi-body acoustic scattering problems involving rotationally symmetric scatterers*, Computers & Mathematics with Applications, 69 (2015), pp. 304 – 318.

[11] R. W. Hockney, *The communication challenge for MPP: Intel Paragon and Meiko CS-2*, Parallel Comput., 20 (1994), p. 389–398.

[12] L. Hörmander, *Fourier integral operators. i*, Acta Math., 127 (1971), pp. 79–183.

[13] Y. Li and H. Yang, *Interpolative butterfly factorization*, SIAM Journal on Scientific Computing, 39 (2017), pp. A503–A531.

[14] Y. Li, H. Yang, E. R. Martin, K. L. Ho, and L. Ying, *Butterfly factorization*, Multiscale Modeling & Simulation, 13 (2015), pp. 714–732.

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

[16] Y. Li, H. Yang, and L. Ying, *Multidimensional butterfly factorization*, Applied and Computational Harmonic Analysis, (2017).

[17] E. Liberty, F. Woolfe, P.-G. Martinsson, V. Rokhlin, and M. Tygert, *Randomized algorithms for the low-rank approximation of matrices*, Proceedings of the National Academy of Sciences, 104 (2007), pp. 20167–20172.

[18] L. Lin, J. Lu, and L. Ying, *Fast construction of hierarchical matrix representation from matrix–vector multiplication*, Journal of Computational Physics, 230 (2011), pp. 4071 – 4087, https://doi.org/https://doi.org/10.1016/j.jcp.2011.02.033.

[19] Y. Liu, P. Ghysels, L. Claus, and X. S. Li, *Sparse approximate multifrontal factorization with butterfly compression for high frequency wave equations*, 2020, https://arxiv.org/abs/2007.00202.

[20] 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 (2017), pp. 1179–1183.

[21] Y. Liu and H. Yang, *A hierarchical butterfly LU preconditioner for two-dimensional electromagnetic scattering problems involving open surfaces*, Journal of Computational Physics, 401 (2020), p. 109014.

[22] P. Martinsson and V. Rokhlin, *A fast direct solver for scattering problems involving elongated structures*, Journal of Computational Physics, 221 (2007), pp. 288 – 302, https://doi.org/https://doi.org/10.1016/j.jcp.2006.06.037.

[23] P. G. Martinsson and V. Rokhlin, *A fast direct solver for boundary integral equations in two dimensions*, Journal of Computational Physics, 205 (2005), pp. 1–23.

[24] E. Michielssen and A. Boag, *Multilevel evaluation of electromagnetic fields for the rapid solution of scattering problems*, Microwave and Optical Technology Letters, 7 (1994), pp. 790–795.

[25] E. Michielssen and A. Boag, *A multilevel matrix decomposition algorithm for analyzing*

scattering from large structures, IEEE Transactions on Antennas and Propagation, 44 (1996), pp. 1086–1093.

[26] E. Michielssen, A. Boag, and W. C. Chew, *Scattering from elongated objects: direct solution in $O(Nlog^2N)$ operations*, IEE Proceedings - Microwaves, Antennas and Propagation, 143 (1996), pp. 277–283.

[27] M. O'Neil, F. Woolfe, and V. Rokhlin, *An algorithm for the rapid evaluation of special function transforms*, Applied and Computational Harmonic Analysis, 28 (2010), pp. 203 – 226. Special Issue on Continuous Wavelet Transform in Memory of Jean Morlet, Part I.

[28] Q. Pang, K. L. Ho, and H. Yang, *Interpolative decomposition butterfly factorization*, SIAM Journal on Scientific Computing, 42 (2020), pp. A1097–A1115, https://doi.org/10.1137/19M1294873.

[29] J. Poulson, L. Demanet, N. Maxwell, and L. Ying, *A parallel butterfly algorithm*, SIAM Journal on Scientific Computing, 36 (2014), pp. C49–C65.

[30] J. M. Tamayo, A. Heldring, and J. M. Rius, *Multilevel adaptive cross approximation (MLACA)*, IEEE Transactions on Antennas and Propagation, 59 (2011), pp. 4600–4608.

[31] M. Tygert, *Fast algorithms for spherical harmonic expansions, III*, Journal of Computational Physics, 229 (2010), pp. 6181 – 6192.

[32] H. Yang, *A unified framework for oscillatory integral transforms: When to use NUFFT or butterfly factorization?*, Journal of Computational Physics, 388 (2019), pp. 103 – 122.

[33] L. Ying, *Sparse Fourier transform via butterfly algorithm*, SIAM Journal on Scientific Computing, 31 (2009), pp. 1678–1694.