SIAM J. MATRIX ANAL. APPL.
Vol. 41, No. 4, pp. 1704–1725

HYBRID MATRIX COMPRESSION FOR HIGH-FREQUENCY PROBLEMS*

STEFFEN BÖRM† AND CHRISTINA BÖRST†

Abstract. Boundary element methods for the Helmholtz equation lead to large dense matrices that can only be handled if efficient compression techniques are used. Directional compression techniques can reach good compression rates even for high-frequency problems. Currently there are two approaches to directional compression: Analytic methods approximate the kernel function, while algebraic methods approximate submatrices. Analytic methods are quite fast and proven to be robust, while algebraic methods yield significantly better compression rates. We present a hybrid method that combines the speed and reliability of analytic methods with the good compression rates of algebraic methods.

Key words. hierarchical matrix, Helmholtz equation, $\mathcal{DH}^2$-matrix, high frequency, matrix compression, data-sparse approximation

AMS subject classifications. 35J05, 65F55, 65N38

DOI. 10.1137/19M124280X

1. Introduction.

We consider the Helmholtz single-layer potential operator

$$\mathcal{G}[u](x) := \int_\Omega g(x,y)u(y)\,dy,$$

where $\Omega \subseteq \mathbb{R}^3$ is a surface and

$$g(x,y) = \frac{\exp(\mathbf{i}\kappa\|x-y\|)}{4\pi\|x-y\|} \tag{1.1}$$

denotes the Helmholtz kernel function with the wave number $\kappa \in \mathbb{R}_{\geq 0}$.

Applying a standard Galerkin discretization scheme with a finite element basis $(\varphi_i)_{i\in\mathcal{I}}$ leads to the stiffness matrix $G \in \mathbb{C}^{\mathcal{I}\times\mathcal{I}}$ given by

$$g_{ij} = \int_\Omega \varphi_i(x) \int_\Omega g(x,y)\varphi_j(y)\,dy\,dx \qquad \text{for all } i,j \in \mathcal{I}, \tag{1.2}$$

where we assume that the basis functions are sufficiently smooth to ensure that the integrals are well defined even if the supports overlap, e.g., for $i = j$. Due to $g(x,y) \neq 0$ for all $x \neq y$, the matrix G is not sparse and therefore requires special handling if we want to construct an efficient algorithm.

Standard techniques, like fast multipole expansions [20, 14], panel clustering [17, 22], or hierarchical matrices [15, 16, 12], rely on local low-rank approximations of the matrix. In the case of the high-frequency Helmholtz equation, e.g., if the product of the wave number κ and the mesh width h is bounded but not particularly small, these techniques can no longer be applied since the local ranks become too large. This situation frequently appears in engineering applications.

*Received by the editors February 5, 2019; accepted for publication (in revised form) by L. Grasedyck August 26, 2020; published electronically November 9, 2020.
https://doi.org/10.1137/19M124280X
†Department of Mathematics, Christian-Albrechts-Universität zu Kiel, Kiel 24098, Germany (boerm@math.uni-kiel.de, boerst@math.uni-kiel.de).

The *fast multipole method* can be generalized to handle this problem by employing a special expansion that leads to operators that can be diagonalized and therefore evaluated efficiently [21, 13, 1, 2].

The *butterfly method* (also known as multilevel matrix decomposition algorithms) [19] achieves a similar goal by using permutations and block-diagonal transformations in a pattern closely related to the fast Fourier transformation algorithm.

Directional methods [7, 9, 18, 3] take advantage of the fact that the Helmholtz kernel (1.1) can be written as a product of a plane wave and a function that is smooth inside a conical domain. Replacing this smooth function by a suitable approximation results in fast summation schemes.

We will focus on directional methods since they can be applied in a more general setting than the fast multipole expansions based on special functions and since they offer the chance of achieving better compression to $\mathcal{O}(n \log n)$ coefficients compared to $\mathcal{O}(n \log^2 n)$ required by the butterfly scheme [19].

In particular, we will work with *directional* $\mathcal{H}^2$*-matrices* (abbreviated $\mathcal{DH}^2$-matrices), the algebraic counterparts of the directional approximation schemes used in [7, 9, 18]. Our starting point is the directional Chebyshev approximation scheme introduced in [18] and analyzed in [6]. While this approach is fast and proven to be reliable, the resulting ranks are quite large, and this leads to unattractive storage requirements.

We can solve this problem by applying an algebraic recompression algorithm that starts with the $\mathcal{DH}^2$-matrix constructed by interpolation and uses nested orthogonal projections and singular value decompositions (SVD) to significantly reduce the rank. This algorithm is based on the general $\mathcal{DH}^2$-matrix compression algorithm introduced in [5] but takes advantage of the previous approximation in order to significantly reduce the computational work to $\mathcal{O}(nk^3 \log n)$ in the high-frequency case; cf. Theorem 4.6 and Remark 4.7.

Compared to the closely related algorithm presented in [18], our algorithm compresses the entire $\mathcal{DH}^2$-matrix structure instead of just the coupling matrices, and the orthogonal projections applied in the recompression algorithm allow us to obtain straightforward estimates for the compression error.

Compared to the algorithm presented in [3], our approach has better stability properties owing to the results of [6] for the interpolation scheme and the orthogonal projections employed in [5] for the recompression, and it can be expected to yield better compression rates since it uses an $\mathcal{H}^2$-matrix representation for low-frequency clusters, while the algorithm of [3] relies on the slightly less efficient $\mathcal{H}$-matrices.

2. Directional $\mathcal{H}^2$-matrices.

Hierarchical matrix methods are based on decompositions of the matrix G into submatrices that can be approximated by factorized low-rank matrices. In our case, we follow the directional interpolation technique described in [18] and translate the resulting compressed representation into an algebraic definition that can be applied in more general situations.

In order to describe the decomposition into submatrices, we first introduce a hierarchy of subsets of the index set $\mathcal{I}$ corresponding to the *box trees* used, e.g., in fast multipole methods.

DEFINITION 2.1 (cluster tree). *Let $\mathcal{T}$ be a labeled tree such that the label $\hat{t}$ of each node $t \in \mathcal{T}$ is a subset of the index set $\mathcal{I}$. We call $\mathcal{T}$ a* cluster tree *for $\mathcal{I}$ if*

- *the root $r \in \mathcal{T}$ is labeled $\hat{r} = \mathcal{I}$;*
- *the index sets of siblings are disjoint, i.e.,*

$$t_1 \neq t_2 \Longrightarrow \hat{t}_1 \cap \hat{t}_2 = \emptyset \qquad \text{for all } t \in \mathcal{T},\ t_1, t_2 \in \operatorname{chil}(t);$$

- *the index sets of a cluster's children are a partition of their parent's index set, i.e.,*

$$\hat{t} = \bigcup_{t' \in \operatorname{chil}(t)} \hat{t}' \qquad \text{for all } t \in \mathcal{T} \text{ with } \operatorname{chil}(t) \neq \emptyset.$$

A cluster tree for $\mathcal{I}$ is usually denoted by $\mathcal{T}_\mathcal{I}$. Its nodes are called clusters. *We denote the* set of leaves *of $\mathcal{T}_\mathcal{I}$ by $\mathcal{L}_\mathcal{I} := \{t \in \mathcal{T}_\mathcal{I} \ : \ \operatorname{chil}(t) = \emptyset\}$.*

A cluster tree $\mathcal{T}_\mathcal{I}$ can be split into levels: We let $\mathcal{T}_\mathcal{I}^{(0)}$ be the set containing only the root of $\mathcal{T}_\mathcal{I}$ and define

$$\mathcal{T}_\mathcal{I}^{(\ell)} := \{t' \in \mathcal{T}_\mathcal{I} \ : \ t' \in \operatorname{chil}(t) \text{ for a } t \in \mathcal{T}_\mathcal{I}^{(\ell-1)}\} \qquad \text{for all } \ell \in \mathbb{N}.$$

For each cluster $t \in \mathcal{T}_\mathcal{I}$, there is exactly one $\ell \in \mathbb{N}_0$ such that $t \in \mathcal{T}_\mathcal{I}^{(\ell)}$. We call this the *level number* of t and denote it by $\operatorname{level}(t) = \ell$. The maximal level

$$p_\mathcal{I} := \max\{\operatorname{level}(t) \ : \ t \in \mathcal{T}_\mathcal{I}\}$$

is called the *depth* of the cluster tree.

Pairs of clusters (t, s) correspond to subsets $\hat{t} \times \hat{s}$ of $\mathcal{I} \times \mathcal{I}$, i.e., to submatrices of $G \in \mathbb{C}^{\mathcal{I} \times \mathcal{I}}$. These pairs inherit the hierarchical structure provided by the cluster tree.

In order to approximate $G|_{\hat{t} \times \hat{s}}$, the directional interpolation approach uses axis-parallel *bounding boxes* $B_t, B_s \subseteq \mathbb{R}^3$ such that

$$\operatorname{supp}(\varphi_i) \subseteq B_t, \qquad \operatorname{supp}(\varphi_j) \subseteq B_s \qquad \text{for all } i \in \hat{t},\ j \in \hat{s},$$

and constructs an approximation $\tilde{g}_{ts}$ of $g|_{B_t \times B_s}$. Discretizing $\tilde{g}_{ts}$ then gives rise to an approximation of the submatrix $G|_{\hat{t} \times \hat{s}}$.

For large wave numbers κ, the function $g|_{B_t \times B_s}$ cannot be expected to be smooth, so we cannot apply interpolation directly. This problem can be solved by *directional* interpolation [7, 9, 18]: We choose a direction $c \in \mathbb{R}^3$ and split the function g into a plane wave and a remainder term; i.e., we use

$$\begin{aligned} g(x,y) &= \exp(\mathbf{i}\kappa\langle x - y, c\rangle) \frac{\exp(\mathbf{i}\kappa(\|x-y\| - \langle x - y, c\rangle))}{4\pi\|x-y\|} \\ &= \exp(\mathbf{i}\kappa\langle x - y, c\rangle) g_c(x,y), \end{aligned}$$

where the remainder is defined by

$$g_c(x,y) = \frac{\exp(\mathbf{i}\kappa(\|x-y\| - \langle x - y, c\rangle))}{4\pi\|x-y\|}.$$

This function is smooth [6] and can therefore be interpolated by polynomials if the following *admissibility conditions* hold:

$$\kappa \left\| \frac{m_t - m_s}{\|m_t - m_s\|} - c \right\| \leq \frac{\eta_1}{\max\{\operatorname{diam}(B_t), \operatorname{diam}(B_s)\}}, \tag{2.1a}$$

$$\max\{\operatorname{diam}(B_t), \operatorname{diam}(B_s)\} \leq \eta_2 \operatorname{dist}(B_t, B_s), \tag{2.1b}$$

$$\kappa \max\{\operatorname{diam}(B_t)^2, \operatorname{diam}(B_s)^2\} \leq \eta_2 \operatorname{dist}(B_t, B_s), \tag{2.1c}$$

where $m_t \in B_t$ and $m_s \in B_s$ denote the midpoints of the boxes and $\eta_1, \eta_2 \in \mathbb{R}_{>0}$ are chosen to strike a balance between fast convergence (if both parameters are small)

and low computational cost (if both parameters are large). Condition (2.1a) ensures that c is sufficiently close to the direction from the midpoint m_s to the midpoint m_t, condition (2.1c) allows us to extend this property to directions from any point $y \in B_s$ to any point $x \in B_t$ [6, Lemma 3.9], while condition (2.1b) is required to keep admissible blocks sufficiently far away from the singularity at $x = y$.

Due to [6, Corollary 3.14], the interpolating polynomial

$$\tilde{g}_{c,ts}(x,y) = \sum_{\nu,\mu=1}^{k} \mathcal{L}_{t,\nu}(x) g_c(\xi_{t,\nu}, \xi_{s,\mu}) \mathcal{L}_{s,\mu}(y)$$

converges exponentially to g_c in $B_t \times B_s$, and the error is bounded independently of the wave number κ. Here $(\xi_{t,\nu})_{\nu=1}^k$ and $(\xi_{s,\mu})_{\mu=1}^k$ are families of tensor interpolation points in B_t and B_s, while $(\mathcal{L}_{t,\nu})_{\nu=1}^k$ and $(\mathcal{L}_{s,\mu})_{\mu=1}^k$ are the corresponding families of tensor Lagrange polynomials.

Multiplying by the plane wave, we obtain

$$\begin{aligned}
g(x,y) &= \exp(\mathbf{i}\kappa\langle x-y, c\rangle) g_c(x,y) \\
&\approx \exp(\mathbf{i}\kappa\langle x-y, c\rangle) \sum_{\nu,\mu=1}^{k} \mathcal{L}_{t,\nu}(x) g_c(\xi_{t,\nu}, \xi_{s,\mu}) \mathcal{L}_{s,\mu}(x) \\
&= \sum_{\nu,\mu=1}^{k} \exp(\mathbf{i}\kappa\langle x, c\rangle) \mathcal{L}_{t,\nu}(x) g_c(\xi_{t,\nu}, \xi_{s,\mu}) \overline{\exp(\mathbf{i}\kappa\langle y, c\rangle) \mathcal{L}_{s,\mu}(y)} \\
&= \sum_{\nu,\mu=1}^{k} \mathcal{L}_{tc,\nu}(x) g_c(\xi_{t,\nu}, \xi_{s,\mu}) \overline{\mathcal{L}_{sc,\mu}(y)} =: \tilde{g}_b(x,y) \qquad \text{for all } x \in B_t,\ y \in B_s
\end{aligned}$$

with the modified Lagrange polynomials

$$\mathcal{L}_{tc,\nu}(x) = \exp(\mathbf{i}\kappa\langle x, c\rangle) \mathcal{L}_{t,\nu}(x), \qquad \mathcal{L}_{sc,\mu}(y) = \exp(\mathbf{i}\kappa\langle y, c\rangle) \mathcal{L}_{s,\mu}(y).$$

Replacing g by $\tilde{g}_b$ in (1.2) yields

$$\begin{aligned}
g_{ij} &\approx \int_\Omega \varphi_i(x) \int_\Omega \tilde{g}_b(x,y) \varphi_j(y)\, dy\, dx \\
&= \sum_{\nu=1}^{k} \sum_{\mu=1}^{k} \underbrace{g_c(\xi_{t,\nu}, \xi_{s,\mu})}_{=: s_{b,\nu\mu}} \underbrace{\int_\Omega \varphi_i(x) \mathcal{L}_{tc,\nu}(x)\, dx}_{=: v_{tc,i\nu}} \underbrace{\int_\Omega \varphi_j(y) \overline{\mathcal{L}_{sc,\mu}(y)}\, dy}_{\overline{v_{sc,j\mu}}} \\
&= \sum_{\nu=1}^{k} \sum_{\mu=1}^{k} s_{b,\nu\mu} v_{tc,i\nu} \overline{v_{sc,j\mu}} = (V_{tc} S_b V_{sc}^*)_{ij} \qquad \text{for all } i \in \hat{t},\ j \in \hat{s}
\end{aligned}$$

with matrices $V_{tc} \in \mathbb{C}^{\hat{t} \times k}$, $V_{sc} \in \mathbb{C}^{\hat{s} \times k}$, and $S_b \in \mathbb{C}^{k \times k}$. This is a factorized low-rank approximation,

$$G|_{\hat{t} \times \hat{s}} \approx V_{tc} S_b V_{sc}^*, \tag{2.2}$$

of the submatrix $G|_{\hat{t} \times \hat{s}}$.

Since the matrix G itself does not satisfy the conditions (2.1), we have to split it into submatrices, and experiments show that the number of submatrices grows rapidly as the problem size increases. Storing the matrices $(V_{tc})_{t \in \mathcal{T}_\mathcal{I}}$ for all clusters $t \in \mathcal{T}_\mathcal{I}$ would lead to quadratic complexity and is therefore unattractive. Fortunately, we

can take advantage of the hierarchical structure of the cluster tree if we organize the directions c accordingly.

DEFINITION 2.2 (directions). *Let $(\mathcal{D}_\ell)_{\ell=0}^{p_\mathcal{I}}$ be a family of finite subsets of $\mathbb{R}^3$. It is called a* family of directions *if*

$$\|c\| = 1 \vee c = 0 \qquad \text{for all } c \in \mathcal{D}_\ell,\ \ell \in [0 : p_\mathcal{I}].$$

Here the special case $c = 0$ is included to allow us to treat the low-frequency case that does not require us to split off a plane wave.

Given a family of directions, we fix a family $(\mathrm{sd}_\ell)_{\ell=0}^{p_\mathcal{I}-1}$ of mappings $\mathrm{sd}_\ell : \mathcal{D}_\ell \to \mathcal{D}_{\ell+1}$ such that

$$\|c - \mathrm{sd}_\ell(c)\| \leq \|c - \tilde{c}\| \qquad \text{for all } c \in \mathcal{D}_\ell,\ \tilde{c} \in \mathcal{D}_{\ell+1},\ \ell \in [0 : p_\mathcal{I} - 1].$$

Given a cluster tree $\mathcal{T}_\mathcal{I}$, we write

$$\mathcal{D}_t := \mathcal{D}_{\mathrm{level}(t)}, \qquad \mathrm{sd}_t(c) := \mathrm{sd}_{\mathrm{level}(t)}(c) \qquad \text{for all } t \in \mathcal{T}_\mathcal{I},\ c \in \mathcal{D}_{\mathrm{level}(t)}.$$

In order to satisfy the admissibility condition (2.1a), we use sets of directions that are sufficiently large to approximate *any* direction; i.e., we require

$$\min\{\|y - c\| \ :\ c \in \mathcal{D}_t\} \leq \frac{\eta_1}{\kappa \operatorname{diam}(B_t)} \quad \text{for all } t \in \mathcal{T}_\mathcal{I},\ y \in \mathbb{R}^3 \text{ with } \|y\| = 1. \tag{2.3}$$

Since the size of clusters decreases as the level grows, this requirement means that large clusters will require more directions than small clusters.

Remark 2.3 (construction of directions). In our implementation, we construct sets of directions satisfying (2.3) as follows: For a level $\ell \in [0 : p_\mathcal{I}]$, we compute the maximal diameter d_ℓ of all bounding boxes B_t associated with clusters $t \in \mathcal{T}_\mathcal{I}^{(\ell)}$ on this level. If $\kappa d_\ell \leq \eta_1$ holds, we let $\mathcal{D}_\ell = \{0\}$; i.e., we use no directional approximation in the low-frequency case.

Otherwise, i.e., if $\kappa d_\ell > \eta_1$, we let $m := \lceil \sqrt{2}\kappa d_\ell / \eta_1 \rceil$ and split each side of the unit cube $[0,1]^3$ into m^2 squares of width $2/m$ and diameter $2\sqrt{2}/m$. Since the cube has six sides, we have a total of $6m^2 \in \mathcal{O}(\kappa^2 d_\ell^2)$ such squares. We denote the centers of the squares by $\tilde{c}_\iota$ and their projections to the unit sphere by $c_\iota := \tilde{c}_\iota / \|\tilde{c}_\iota\|_2$ for $\iota \in [1 : 6m^2]$. We let $\mathcal{D}_\ell := \{c_\iota \ :\ \iota \in [1 : 6m^2]\}$. For every unit vector $y \in \mathbb{R}^3$, there is a point $\tilde{y}$ on the surface of the unit cube with $y = \tilde{y}/\|\tilde{y}\|_2$. Since the surface grid is sufficiently fine, we can find $\iota \in [1 : 6m^2]$ with $\|\tilde{y} - \tilde{c}_\iota\|_2 \leq \sqrt{2}/m$, and the projection ensures that

$$\|y - c_\iota\|_2 \leq \|\tilde{y} - \tilde{c}_\iota\|_2 \leq \frac{\sqrt{2}}{m} \leq \frac{\eta_1}{\kappa d_\ell} \leq \frac{\eta_1}{\kappa \operatorname{diam}(B_t)} \qquad \text{for all } t \in \mathcal{T}_\mathcal{I}^{(\ell)};$$

i.e., (2.3) holds. By this construction, the set $\mathcal{D}_\ell$ is sufficiently large to contain approximations for *any* unit vector but still small enough to satisfy the assumption (4.1) required for our complexity analysis.

Due to (2.3), we only have to satisfy the conditions (2.1b) and (2.1c) and can then find a direction $c_{ts} \in \mathcal{D}_t = \mathcal{D}_s$ that satisfies the first condition (2.1a): For $t, s \in \mathcal{T}_\mathcal{I}^{(\ell)}$,

we let $c_{ts} \in \mathcal{D}_\ell$ be a best approximation of the direction from the midpoint m_s of the source box B_s to midpoint m_t of the target box B_t, i.e.,

$$\left\| \frac{m_t - m_s}{\|m_t - m_s\|_2} - c_{ts} \right\|_2 \leq \left\| \frac{m_t - m_s}{\|m_t - m_s\|_2} - \tilde{c} \right\|_2 \qquad \text{for all } \tilde{c} \in \mathcal{D}_\ell.$$

If $m_t = m_s$, the admissibility condition (2.1b) is violated and we can choose any direction for c_{ts}. This leaves us with the task of splitting the matrix G into submatrices $G|_{\hat{t}\times\hat{s}}$ such that B_t and B_s satisfy the admissibility conditions (2.1b) and (2.1c). A decomposition with the minimal necessary number of submatrices can be constructed by a recursive procedure that again gives rise to a tree structure and inductively ensures that $\operatorname{level}(t) = \operatorname{level}(s)$, so that the directions c_{ts} are well defined.

DEFINITION 2.4 (block tree). *Let $\mathcal{T}_\mathcal{I}$ be a cluster tree for the index set $\mathcal{I}$ with root $r_\mathcal{I}$, and let $(\mathcal{D}_\ell)_{\ell=0}^{p_\mathcal{I}}$ be a family of directions.*

A tree $\mathcal{T}$ is called a block tree *for $\mathcal{T}_\mathcal{I}$ if*

- *for each $b \in \mathcal{T}$ there are $t, s \in \mathcal{T}_\mathcal{I}$ such that $b = (t, s, c_{ts})$;*
- *the root $r \in \mathcal{T}$ satisfies $r = (r_\mathcal{I}, r_\mathcal{I}, c_{r_\mathcal{I} r_\mathcal{I}})$;*
- *for each $b = (t, s, c_{ts}) \in \mathcal{T}$ we have*

$$\operatorname{chil}(b) \neq \emptyset \Longrightarrow \operatorname{chil}(b) = \{(t', s', c_{t's'}) \ : \ t' \in \operatorname{chil}(t),\ s' \in \operatorname{chil}(s)\}. \tag{2.4}$$

A block tree for $\mathcal{T}_\mathcal{I}$ is usually denoted by $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$. Its nodes are called blocks. *We denote the set of leaves of $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ by $\mathcal{L}_{\mathcal{I}\times\mathcal{I}} := \{b \in \mathcal{T}_{\mathcal{I}\times\mathcal{I}} \ : \ \operatorname{chil}(b) = \emptyset\}$.*

The leaves of a block tree define a disjoint partition of the index set $\mathcal{I} \times \mathcal{I}$, i.e., a decomposition of the matrix $G \in \mathbb{C}^{\mathcal{I}\times\mathcal{I}}$ into submatrices $G|_{\hat{t}\times\hat{s}}$ with $(t, s, c) \in \mathcal{L}_{\mathcal{I}\times\mathcal{I}}$.

We can construct a block tree with the minimal number of blocks by a simple recursion: Starting with the root, we check whether a block is admissible. If it is, we make it an admissible leaf and represent the corresponding submatrix in the factorized form (2.2). Otherwise, we consider its children given by (2.4). If there are no children, i.e., if $\operatorname{chil}(t)$ or $\operatorname{chil}(s)$ are empty, we have found an inadmissible leaf and store the submatrix directly, i.e., as a two-dimensional array.

While the approximation (2.2) reduces the amount of storage required for one block to k^2, we still have to store the matrices $(V_{tc})_{t\in\mathcal{T}_\mathcal{I}, c\in\mathcal{D}_t}$, and in the high-frequency case the storage requirements for these matrices grow at least quadratically with the problem size: If we have $\kappa^2 \sim n$, doubling the matrix dimension means multiplying κ by a factor of $\sqrt{2}$. Constructing directions as in Remark 2.3, the splitting parameter m also grows by a factor of approximately $\sqrt{2}$, and the number of directions is therefore doubled as well. On a given level $\ell \in [0 : p_\mathcal{I}]$, storing V_{tc} for all $t \in \mathcal{T}_\mathcal{I}$ with a *fixed* direction c requires $\mathcal{O}(nk)$ coefficients, and since we have $\mathcal{O}(n)$ directions, we even end up with $\mathcal{O}(n^2 k)$ coefficients *per level*.

In order to solve this problem, we take advantage of the requirement (2.3): Given a cluster $t \in \mathcal{T}_\mathcal{I}$, a direction $c \in \mathcal{D}_t$, and one of its children $t' \in \operatorname{chil}(t)$, we can find a direction $c' := \operatorname{sd}_\ell(c) \in \mathcal{D}_{t'}$ that approximates c reasonably well. This property allows us to reduce the storage requirements for the matrices $(V_{tc})_{t\in\mathcal{T}_\mathcal{I}, c\in\mathcal{D}_t}$ as follows: Since $\|c - c'\|_2$ is small, the function

$$x \mapsto \exp(-\mathbf{i}\kappa\langle x, c'\rangle)\mathcal{L}_{tc,\nu}(x) = \exp(\mathbf{i}\kappa\langle x, c - c'\rangle)\mathcal{L}_{t,\nu}(x)$$

is smooth and can therefore be interpolated in $B_{t'}$. We find

$$\begin{aligned}\mathcal{L}_{tc,\nu}(x) &= \exp(\mathbf{i}\kappa\langle x, c\rangle)\mathcal{L}_{t,\nu}(x) = \exp(\mathbf{i}\kappa\langle x, c'\rangle)\exp(\mathbf{i}\kappa\langle x, c-c'\rangle)\mathcal{L}_{t,\nu}(x)\\ &\approx \exp(\mathbf{i}\kappa\langle x, c'\rangle)\sum_{\nu'=1}^{k}\underbrace{\exp(\mathbf{i}\kappa\langle \xi_{t',\nu'}, c-c'\rangle)\mathcal{L}_{t,\nu}(\xi_{t',\nu'})}_{=:e_{t'c,\nu'\nu}}\mathcal{L}_{t',\nu'}(x)\\ &= \sum_{\nu'=1}^{k} e_{t'c,\nu'\nu}\mathcal{L}_{t'c',\nu'}(x).\end{aligned}$$

This approach immediately yields

$$v_{tc,i\nu} = \int_\Omega \varphi_i(x)\mathcal{L}_{tc,\nu}(x)\,dx \approx \sum_{\nu'=1}^{k} e_{t'c,\nu'\nu}\int_\Omega \varphi_i(x)\mathcal{L}_{t'c',\nu'}(x)\,dx = (V_{t'c'}E_{t'c})_{i\nu}$$

for all $i \in \hat{t}'$ and $\nu \in [1:k]$, which is equivalent to

$$V_{tc}|_{\hat{t}'\times k} \approx V_{t'c'}E_{t'c}. \tag{2.5}$$

Instead of storing V_{tc}, we can just store small $k \times k$ matrices $E_{t'c}$ for all $t' \in \operatorname{chil}(t)$, thus reducing the storage requirements from $(\#\hat{t})k$ to $\mathcal{O}(k^2)$. This approach implies using the right-hand side of (2.5) to *define* the left-hand side.

DEFINITION 2.5 (directional cluster basis). *Let $k \in \mathbb{N}$, and let $V = (V_{tc})_{t\in\mathcal{T}_\mathcal{I}, c\in\mathcal{D}_t}$ be a family of matrices. We call it a* directional cluster basis *if*

- *$V_{tc} \in \mathbb{C}^{\hat{t}\times k}$ for all $t \in \mathcal{T}_\mathcal{I}$ and $c \in \mathcal{D}_t$;*
- *there is a family $E = (E_{t'c})_{t\in\mathcal{T}_\mathcal{I}, t'\in\operatorname{chil}(t), c\in\mathcal{D}_t}$ such that*

$$V_{tc}|_{\hat{t}'\times k} = V_{t'c'}E_{t'c} \quad \text{for all } t \in \mathcal{T}_\mathcal{I},\ t' \in \operatorname{chil}(t),\ c \in \mathcal{D}_t,\ c' = \operatorname{sd}_t(c). \tag{2.6}$$

The elements of the family E are called transfer matrices *for the directional cluster basis V, and k is called its* rank.

Remark 2.6 (notation). The notation "$E_{t'c}$" for the transfer matrices (instead of something like "$E_{t'tc'c}$" listing all parameters) for the matrices is justified since the parent $t \in \mathcal{T}_\mathcal{I}$ is uniquely determined by $t' \in \mathcal{T}_\mathcal{I}$ due to the tree structure, and the direction $c' = \operatorname{sd}_t(c)$ is uniquely determined by $c \in \mathcal{D}_t$ due to our Definition 2.2.

We can now define the class of matrices that is the subject of this article: Since the leaves $\mathcal{L}_{\mathcal{I}\times\mathcal{I}}$ of the block tree correspond to a partition of the matrix G, we have to represent each of the submatrices $G|_{\hat{t}\times\hat{s}}$ for $b = (t, s, c) \in \mathcal{L}_{\mathcal{I}\times\mathcal{I}}$. Those blocks that satisfy the admissibility conditions (2.1) can be approximated in the form (2.2). These matrices are called *admissible* and are collected in a subset:

$$\mathcal{L}^+_{\mathcal{I}\times\mathcal{I}} := \{b \in \mathcal{L}_{\mathcal{I}\times\mathcal{I}} \ :\ b \text{ is admissible}\}.$$

The remaining blocks are called *inadmissible* and are collected in the set

$$\mathcal{L}^-_{\mathcal{I}\times\mathcal{I}} := \mathcal{L}_{\mathcal{I}\times\mathcal{I}} \setminus \mathcal{L}^+_{\mathcal{I}\times\mathcal{I}}.$$

These matrices are stored as simple two-dimensional arrays without any compression.

DEFINITION 2.7 (directional $\mathcal{H}^2$-matrix). *Let V and W be directional cluster bases for $\mathcal{T}_\mathcal{I}$. Let $G \in \mathbb{C}^{\mathcal{I}\times\mathcal{I}}$ be a matrix. We call it a* directional $\mathcal{H}^2$-matrix *(or just*

a $\mathcal{DH}^2$-matrix*) if there are families* $S = (S_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{I}}}$ *such that*

$$G|_{\hat{t}\times\hat{s}} = V_{tc}S_bW_{sc}^* \qquad \text{for all } b = (t,s,c) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{I}}. \tag{2.7}$$

The elements of the family S *are called* coupling matrices*.* V *is called the* row cluster basis*, and* W *is called the* column cluster basis*.*

A $\mathcal{DH}^2$-matrix representation *of a* $\mathcal{DH}^2$*-matrix* G *consists of* V*,* W*,* S*, and the family* $(G|_{\hat{b}})_{b\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{I}}}$ *of* nearfield matrices *corresponding to the inadmissible leaves of* $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$*.*

Under typical assumptions, including that k is fixed independently of κ, since the interpolation error does not depend on κ, it is possible to prove that a $\mathcal{DH}^2$-matrix requires only $\mathcal{O}(nk + \kappa^2k^2\log(n))$ units of storage [5, section 5].

3. Recompression.

3.1. Compression of general matrices. Before we address the recompression of a $\mathcal{DH}^2$-matrix, we briefly recall the compression algorithm for general matrices described in [5].

Let $G \in \mathbb{C}^{\mathcal{I}\times\mathcal{I}}$. We want to approximate the matrix by an *orthogonal projection* since this guarantees optimal stability and best-approximation properties with respect to certain norms.

We call a matrix $X \in \mathbb{C}^{\mathcal{I}\times\mathcal{K}}$ *isometric* if $X^*X = I$ holds. If X is isometric, XX^* is the orthogonal projection into the range of X; i.e., it maps a vector $y \in \mathbb{C}^{\mathcal{I}}$ onto its best approximation $\widetilde{y} := XX^*y$ in this space, and the stability estimate $\|\widetilde{y}\|_2 \leq \|y\|_2$ holds.

We call the cluster bases $(V_{tc})_{t\in\mathcal{T}_{\mathcal{I}},c\in\mathcal{D}_t}$ and $(W_{tc})_{t\in\mathcal{T}_{\mathcal{I}},c\in\mathcal{D}_t}$ *orthogonal* if all matrices are isometric, i.e., if

$$V_{tc}^*V_{tc} = I, \qquad W_{tc}^*W_{tc} = I \qquad \text{holds for all } t \in \mathcal{T}_{\mathcal{I}},\ c \in \mathcal{D}_t.$$

In this case, the optimal coupling matrices with respect to the Frobenius norm (and almost optimal with respect to the spectral norm) can be computed by orthogonal projection using

$$G|_{\hat{t}\times\hat{s}} \approx V_{tc}V_{tc}^*G|_{\hat{t}\times\hat{s}}W_{sc}W_{sc}^* = V_{tc}S_bW_{tc}^* \qquad \text{with} \qquad S_b := V_{tc}^*G|_{\hat{t}\times\hat{s}}W_{sc}.$$

Due to

$$\begin{aligned}\|G|_{\hat{t}\times\hat{s}} - V_{tc}S_bW_{sc}^*\|_F^2 &= \|G|_{\hat{t}\times\hat{s}} - V_{tc}V_{tc}^*G|_{\hat{t}\times\hat{s}}\|_F^2 + \|V_{tc}V_{tc}^*(G|_{\hat{t}\times\hat{s}} - G|_{\hat{t}\times\hat{s}}W_{sc}W_{sc}^*)\|_F^2\\ &\leq \|G|_{\hat{t}\times\hat{s}} - V_{tc}V_{tc}^*G|_{\hat{t}\times\hat{s}}\|_F^2 + \|G|_{\hat{t}\times\hat{s}}^* - W_{sc}W_{sc}^*G|_{\hat{t}\times\hat{s}}^*\|_F^2,\end{aligned}$$

we can focus on the construction of a good row cluster basis since a good column cluster basis can be obtained by applying the same procedure to the adjoint matrix.

By Definition 2.7, the matrix V_{tc} has to be able to approximate the range of all matrices $G|_{\hat{t}\times\hat{s}}$ with $(t,s,c) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{I}}$. We collect the corresponding column clusters in the set

$$\operatorname{row}(t,c) := \{s \in \mathcal{T}_{\mathcal{I}} \ :\ (t,s,c) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{I}}\}.$$

We also have to take the nested structure of the cluster basis into account. Let $t \in \mathcal{T}_{\mathcal{I}}$ with $\operatorname{chil}(t) \neq \emptyset$. For the sake of simplicity, we focus on the case $\#\operatorname{chil}(t) = 2$ and

$\operatorname{chil}(t) = \{t_1, t_2\}$. Assume that isometric matrices $V_{t_1 c_1}$ and $V_{t_2 c_2}$ with $c_1 = \operatorname{sd}_{t_1}(c)$ and $c_2 = \operatorname{sd}_{t_2}(c)$ have already been computed. Due to (2.6), we have

$$V_{tc} = \begin{pmatrix} V_{t_1 c_1} & \\ & V_{t_2 c_2} \end{pmatrix} \widehat{V}_{tc} \qquad \text{with} \qquad \widehat{V}_{tc} := \begin{pmatrix} E_{t_1 c} \\ E_{t_2 c} \end{pmatrix}. \tag{3.1}$$

The error of the orthogonal projection takes the form

$$\|G|_{\hat{t}\times\hat{s}} - V_{tc} V_{tc}^* G|_{\hat{t}\times\hat{s}}\|_F^2 = \left\| G|_{\hat{t}\times\hat{s}} - \begin{pmatrix} V_{t_1 c_1} & \\ & V_{t_2 c_2} \end{pmatrix} \widehat{V}_{tc} \widehat{V}_{tc}^* \begin{pmatrix} V_{t_1 c_1}^* & \\ & V_{t_2 c_2}^* \end{pmatrix} G|_{\hat{t}\times\hat{s}} \right\|_F^2.$$

Since $V_{t_1 c_1}$ and $V_{t_2 c_2}$ are assumed to be isometric, Pythagoras' identity yields

$$\begin{aligned} \|G|_{\hat{t}\times\hat{s}} - V_{tc} V_{tc}^* G|_{\hat{t}\times\hat{s}}\|_F^2 &= \left\| G|_{\hat{t}\times\hat{s}} - \begin{pmatrix} V_{t_1 c_1} & \\ & V_{t_2 c_2} \end{pmatrix} \begin{pmatrix} V_{t_1 c_1}^* & \\ & V_{t_2 c_2}^* \end{pmatrix} G|_{\hat{t}\times\hat{s}} \right\|_F^2 \\ &\quad + \left\| \begin{pmatrix} V_{t_1 c_1} & \\ & V_{t_2 c_2} \end{pmatrix} (I - \widehat{V}_{tc} \widehat{V}_{tc}^*) \begin{pmatrix} V_{t_1 c_1}^* & \\ & V_{t_2 c_2}^* \end{pmatrix} G|_{\hat{t}\times\hat{s}} \right\|_F^2 \\ &= \|G|_{\hat{t}_1\times s} - V_{t_1 c_1} V_{t_1 c_1}^* G|_{\hat{t}_1\times s}\|_F^2 \\ &\quad + \|G|_{\hat{t}_2\times s} - V_{t_2 c_2} V_{t_2 c_2}^* G|_{\hat{t}_2\times s}\|_F^2 \\ &\quad + \left\| (I - \widehat{V}_{tc} \widehat{V}_{tc}^*) \begin{pmatrix} V_{t_1 c_1}^* G|_{\hat{t}_1\times s} \\ V_{t_2 c_2}^* G|_{\hat{t}_2\times s} \end{pmatrix} \right\|_F^2. \end{aligned} \tag{3.2}$$

We can see that the projection error for the cluster t depends on the projection errors for its children t_1 and t_2. Using a straightforward induction, we find that all descendants of t contribute to the error.

This means that our algorithm has to take all ancestors of a cluster t into account when it constructs V_{tc}. We collect these ancestors and the corresponding directions in the sets

$$\operatorname{anc}(t,c) := \begin{cases} \{(t,c)\} & \text{if } t \text{ is the root of } \mathcal{T}_{\mathcal{I}}, \\ \{(t,c)\} \cup \bigcup_{c^+ \in \operatorname{sd}_{t^+}^{-1}(\{c\})} \operatorname{anc}(t^+, c^+) & \text{if } t \text{ is a child of } t^+ \in \mathcal{T}_{\mathcal{I}} \end{cases} \tag{3.3}$$

for all $t \in \mathcal{T}_{\mathcal{I}}$ and $c \in \mathcal{D}_t$. The mapping sd_{t^+} is not invertible since the parent cluster frequently is associated with more directions than its child. This is the reason there can be multiple $c^+ \in \mathcal{D}_{t^+}$ with $\operatorname{sd}_{t^+}(c^+) = c$ in (3.3). We have to find V_{tc} such that

$$G|_{\hat{t}\times\hat{s}} \approx V_{tc} V_{tc}^* G|_{\hat{t}\times\hat{s}} \qquad \text{for all } s \in \operatorname{row}(\tilde{t},\tilde{c}),\ (\tilde{t},\tilde{c}) \in \operatorname{anc}(t,c). \tag{3.4}$$

Due to Definition 2.4, the index sets of the clusters in

$$\operatorname{row}^+(t,c) := \bigcup_{(\tilde{t},\tilde{c}) \in \operatorname{anc}(t,c)} \operatorname{row}(\tilde{t},\tilde{c})$$

are disjoint, and we can introduce

$$\mathcal{R}_{tc} := \bigcup_{s \in \operatorname{row}^+(t,c)} \hat{s}, \qquad G_{tc} := G|_{\hat{t}\times\mathcal{R}_{tc}} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}},\ c \in \mathcal{D}_t$$

in order to rewrite (3.4) in the form

$$G_{tc} \approx V_{tc} V_{tc}^* G_{tc}.$$

If t is a leaf cluster, we can directly find the optimal approximation by computing the SVD of G_{tc} and using the first k left singular vectors as the columns of the matrix V_{tc}: The SVD yields an orthonormal basis $(v_i)_{i=1}^{\tau}$ of left singular vectors, an orthonormal

basis $(u_i)_{i=1}^{\tau}$ of right singular vectors, and ordered singular values $\sigma_1 \geq \cdots \geq \sigma_\tau \geq 0$ with

$$G_{tc} = \sum_{i=1}^{\tau} v_i \sigma_i u_i^*,$$

where $\tau = \#\hat{t}$. Using this notation, it is an easy task to find the lowest rank $k \in [0 : \tau]$ such that the approximation

$$\widetilde{G}_{tc} := \sum_{i=1}^{k} v_i \sigma_i u_i^* = V_{tc} V_{tc}^* G_{tc}, \qquad V_{tc} := \begin{pmatrix} v_1 & \dots & v_k \end{pmatrix},$$

still ensures the desired error bound [4, Lemma 5.19]. For the sake of simplicity, we consider only the Frobenius norm case; the spectral norm and relative errors bounds are available with slight adaptations in the choice of k [11, Theorem 2.5.3].

If t is not a leaf cluster, (3.2) indicates that we have to look for $\widehat{V}_{tc}$ such that

$$\widehat{G}_{tc} \approx \widehat{V}_{tc} \widehat{V}_{tc}^* \widehat{G}_{tc}$$

with the matrix

$$\widehat{G}_{tc} := \begin{pmatrix} V_{t_1 c_1}^* & \\ & V_{t_2 c_2}^* \end{pmatrix} G_{tc} \tag{3.5}$$

containing the coefficients for the approximation of G_{tc} in the children's bases. This task can again be solved by computing the SVD of $\widehat{G}_{tc}$, which takes only $\mathcal{O}(k^2 \# \mathcal{R}_{tc})$ operations, and the transfer matrices $E_{t_1 c}$ and $E_{t_2 c}$ can be obtained from $\widehat{V}_{tc}$ by definition (3.1).

3.2. $\mathcal{DH}^2$-recompression. The algorithm presented in the previous section has quadratic complexity if we assume that the numerical ranks k are uniformly bounded (cf. [5, Theorem 17]), since it starts with a dense matrix G represented explicitly by n^2 coefficients. This means that the algorithm is only of theoretical interest, i.e., for investigating whether a given matrix can be approximated at all, but not attractive for real applications with large numbers of degrees of freedom.

In the case of the Helmholtz equation, it has already been proven [6] that directional interpolation provides us with an $\mathcal{DH}^2$-matrix approximation, although the rank of this approximation may be larger than necessary. Our task is therefore only to *recompress* an already compressed $\mathcal{DH}^2$-matrix; we do not have to start from scratch. If we can arrange the algorithm in a way that avoids creating the entire original approximation, we can obtain nearly optimal storage requirements without the need of excessive storage for intermediate results.

Our first step is to take advantage of the $\mathcal{DH}^2$-matrix structure to reduce the complexity of our algorithm. We assume that the original matrix is described by cluster bases $(V_{tc})_{t \in \mathcal{T}_\mathcal{I}, c \in \mathcal{D}_t}$, $(W_{tc})_{t \in \mathcal{T}_\mathcal{I}, c \in \mathcal{D}_t}$ and coupling matrices $(S_b)_{b \in \mathcal{L}^+_{\mathcal{I} \times \mathcal{I}}}$ such that

$$G|_{\hat{t} \times \hat{s}} = V_{tc} S_b W_{sc}^* \qquad \text{for all } b = (t, s, c) \in \mathcal{L}^+_{\mathcal{I} \times \mathcal{I}}.$$

We denote the transfer matrices of the cluster basis $(V_{tc})_{t \in \mathcal{T}_\mathcal{I}, c \in \mathcal{D}_t}$ by $E_{t'c} \in \mathbb{C}^{k \times k}$ for $t \in \mathcal{T}_\mathcal{I}$, $t' \in \operatorname{chil}(t)$, $c \in \mathcal{D}_t$, and $c' = \operatorname{sd}_t(c)$.

Our goal is to find improved row and column cluster bases for the matrix G by the algorithm outlined in section 3.1 but to take advantage of the $\mathcal{DH}^2$-matrix structure of G to reduce the computational work.

We have already seen that we only have to describe an algorithm for computing row bases since applying this algorithm to the adjoint matrix G^* will yield a column basis, as well. We call the improved row basis $(Q_{tc})_{t\in\mathcal{T}_\mathcal{I},c\in\mathcal{D}_t}$, the adaptively chosen rank of Q_{tc} is called k_{tc}, and the transfer matrices are called $(F_{t'c})_{t\in\mathcal{T}_\mathcal{I},t'\in\operatorname{chil}(t),c\in\mathcal{D}_t}$.

In particular, the matrices V_{tc} and W_{tc} are no longer isometric, and ensuring reliable error control requires the use of suitable weight matrices. Since we assume that V_{tc} and W_{tc} result from directional interpolation of constant order, we know that all of these matrices have a fixed number k of columns.

Our approach to speeding up the algorithm of section 3.1 is to obtain a factorized low-rank representation of the matrices G_{tc} required by the compression algorithm that allows us to efficiently compute an improved basis. In particular, we will prove that there are $k\times k$ matrices $\widehat{Z}_{tc}$ for all $t\in\mathcal{T}_\mathcal{I}$, $c\in\mathcal{D}_t$ such that

$$G_{tc} = V_{tc}\widehat{Z}_{tc}^* P_{tc}^* \tag{3.6}$$

holds with an isometric matrix $P_{tc}\in\mathbb{C}^{\mathcal{R}_{tc}\times k}$. Since P_{tc} is isometric, it does not influence the left singular vectors or the nonzero singular values, so we can replace G_{tc} by the skinny matrix $V_{tc}\widehat{Z}_{tc}^*$ in the compression algorithm of section 3.1 and construct Q_{tc} from the left singular vectors of this smaller matrix without changing the result.

Let $t\in\mathcal{T}_\mathcal{I}$, $c\in\mathcal{D}_t$. For the moment, we assume that t is not the root of the cluster tree, i.e., that it has a parent $t^+\in\mathcal{T}_\mathcal{I}$ with $t\in\operatorname{chil}(t^+)$.

We assume that the matrices $\widehat{Z}_{t^+c^+}$ have already been computed for all directions $c^+\in\mathcal{D}_{t^+}$ with $\operatorname{sd}_{t^+}(c^+)=c$, i.e., for all $c^+\in\operatorname{sd}_{t^+}^{-1}(\{c\})$. Let $\gamma := \#\operatorname{sd}_{t^+}^{-1}(\{c\})$ denote the number of directions in $\mathcal{D}_{t^+}$ that get mapped to c, and enumerate these directions as $\operatorname{sd}_{t^+}^{-1}(\{c\}) = \{c_1^+,\ldots,c_\gamma^+\}$. Due to definition (3.3), we have

$$\operatorname{anc}(t,c) = \{(t,c)\}\cup\bigcup_{\iota=1}^{\gamma}\operatorname{anc}(t^+,c_\iota^+).$$

We let $\sigma := \#\operatorname{row}(t,c)$ and $\operatorname{row}(t,c)=\{s_1,\ldots,s_\sigma\}$.

Let now $s\in\operatorname{row}^+(t,c)$. By definition, either we can have $s\in\operatorname{row}(t,c)$ or there is a $\iota\in[1:\gamma]$ such that $s\in\operatorname{row}^+(t^+,c_\iota^+)$. In the first case, we have

$$G|_{\hat{t}\times\hat{s}} = V_{tc}S_bW_{sc}^*,$$

and we collect all of these matrices in an auxiliary matrix:

$$\begin{aligned} H_{tc} &:= \begin{pmatrix} V_{tc}S_{ts_1c}W_{s_1c}^* & \cdots & V_{tc}S_{ts_\sigma c}W_{s_\sigma c}^*\end{pmatrix}\\ &= V_{tc}\underbrace{\begin{pmatrix} S_{ts_1c}W_{s_1c}^* & \cdots & S_{ts_\sigma c}W_{s_\sigma c}^*\end{pmatrix}}_{=:Y_{tc}}. \end{aligned}$$

The matrix Y_{tc} has too many columns for a practical algorithm, so we use the orthogonalization algorithm [4, Algorithm 16], with a straightforward generalization, to find $k\times k$ matrices R_{W,s_ic} and isometric matrices P_{W,s_ic} with

$$W_{s_ic} = P_{W,s_ic}R_{W,s_ic} \qquad \text{for all } i\in[1:\sigma] \tag{3.7}$$

and obtain

$$Y_{tc} = \underbrace{\begin{pmatrix} S_{ts_1c}R^*_{W,s_1c} & \cdots & S_{ts_\sigma c}R^*_{W,s_\sigma c} \end{pmatrix}}_{=:\widehat{Y}_{tc}} \underbrace{\begin{pmatrix} P_{W,s_1c} & & \\ & \ddots & \\ & & P_{W,s_\sigma c} \end{pmatrix}^*}_{=:P^*_{Y,tc}} = \widehat{Y}_{tc}P^*_{Y,tc}.$$

The matrix $\widehat{Y}_{tc}$ is now sufficiently small, and the isometric matrix $P_{Y,tc}$ can later be subsumed in P_{tc}.

In the second case, i.e., if $s \in \operatorname{row}^+(t^+, c_\iota^+)$, we have

$$G|_{\hat{t}\times\hat{s}} = G_{t^+c_\iota^+}|_{\hat{t}\times\hat{s}}.$$

Combining both cases yields

$$G_{tc} = \begin{pmatrix} H_{tc} & G_{t^+c_1^+}|_{\hat{t}\times\mathcal{R}_{t^+c_1^+}} & \cdots & G_{t^+c_\gamma^+}|_{\hat{t}\times\mathcal{R}_{t^+c_\gamma^+}} \end{pmatrix}.$$

Due to our assumption, we have low-rank representations of the form (3.6) at our disposal for $G_{t^+c_1^+}, \ldots, G_{t^+c_\gamma^+}$, and applying (2.6) yields

$$\begin{aligned} G_{tc} &= \begin{pmatrix} H_{tc} & V_{tc_1}|_{\hat{t}'\times k}\widehat{Z}^*_{tc_1}P^*_{tc_1} & \cdots & V_{t^+c_\gamma^+}|_{\hat{t}\times k}\widehat{Z}^*_{t^+c_\gamma^+}P^*_{t^+c_\gamma^+} \end{pmatrix} \\ &= V_{tc}\begin{pmatrix} \widehat{Y}_{tc}P^*_{Y,tc} & E_{tc_1^+}\widehat{Z}^*_{t^+c_1^+}P^*_{t^+c_1^+} & \cdots & E_{tc_\gamma^+}\widehat{Z}^*_{t^+c_\gamma^+}P^*_{t^+c_\gamma^+} \end{pmatrix} \\ &= V_{tc}\underbrace{\begin{pmatrix} \widehat{Y}_{tc} & E_{tc_1^+}\widehat{Z}^*_{t^+c_1^+} & \cdots & E_{tc_\iota^+}\widehat{Z}^*_{t^+c_\gamma^+} \end{pmatrix}}_{=:Z_{tc}} \begin{pmatrix} P^*_{Y,tc} & & & \\ & P^*_{t^+c_1^+} & & \\ & & \ddots & \\ & & & P^*_{t^+c_\gamma^+} \end{pmatrix}. \end{aligned}$$

We compute a skinny QR factorization

$$\widehat{P}_{tc}\widehat{Z}_{tc} = Z^*_{tc}$$

and find

$$G_{tc} = V_{tc}\widehat{Z}^*_{tc}P^*_{tc} \qquad \text{with} \qquad P_{tc} := \begin{pmatrix} P_{Y,tc} & & & \\ & P_{t^+c_1^+} & & \\ & & \ddots & \\ & & & P_{t^+c_\gamma^+} \end{pmatrix}\widehat{P}_{tc}.$$

As a product of two isometric matrices, P_{tc} is again isometric, and since Z_{tc} has only k rows, $\widehat{Z}_{tc}$ is a $k \times k$ matrix. It is important to note that we do not need the matrices $P_{t^+c_\iota^+}$ to compute $\widehat{Z}_{tc}$; we can carry out the entire algorithm without storing any of the isometric matrices.

If t is the root cluster, it has no parent t^+, but we can still proceed as before by setting $\gamma = 0$, i.e., without contributions inherited from the ancestors.

Once we have the *total weight matrices* $\widehat{Z}_{tc} \in \mathbb{C}^{k\times k}$ at our disposal, we can consider the construction of the basis. Since V_{tc} is already the name of the original basis, we use Q_{tc} for the new one. The transfer matrices for Q_{tc} are denoted by $F_{t'c}$.

If t is a leaf, we have to compute the left singular vectors and singular values of the matrix

$$G_{tc} = V_{tc}\widehat{Z}_{tc}^* P_{tc}^*,$$

and this is equivalent to computing these quantities only for the thin matrix $V_{tc}\widehat{Z}_{tc}^*$. We choose a rank k_{tc} for the new basis and use the first k_{tc} left singular vectors as columns of the new basis matrix $Q_{tc} \in \mathbb{C}^{\hat{t}\times k_{tc}}$.

If t is not a leaf, we assume again $\operatorname{chil}(t) = \{t_1, t_2\}$; let $c_1 := \operatorname{sd}_{t_1}(c)$, $c_2 := \operatorname{sd}_{t_2}(c)$; and have to compute the left singular vectors and singular values of the matrix

$$\begin{aligned}\widehat{G}_{tc} &= \begin{pmatrix} Q_{t_1c_1}^* & \\ & Q_{t_2c_2}^* \end{pmatrix} G_{tc} = \begin{pmatrix} Q_{t_1c_1}^* & \\ & Q_{t_2c_2}^* \end{pmatrix} V_{tc}\widehat{Z}_{tc}^* P_{tc}^* \\ &= \begin{pmatrix} Q_{t_1c_1}^* & \\ & Q_{t_2c_2}^* \end{pmatrix} \begin{pmatrix} V_{t_1c_1}E_{t_1c} \\ V_{t_2c_2}E_{t_2c} \end{pmatrix} \widehat{Z}_{tc}^* P_{tc}^* = \begin{pmatrix} Q_{t_1c_1}^* V_{t_1c_1}E_{t_1c} \\ Q_{t_2c_2}^* V_{t_2c_2}E_{t_2c} \end{pmatrix} \widehat{Z}_{tc}^* P_{tc}^*.\end{aligned}$$

In order to prepare this matrix efficiently, we introduce the matrices

$$C_{tc} := Q_{tc}^* V_{tc} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}},\ c \in \mathcal{D}_t, \tag{3.8}$$

which describe the change of basis from V_{tc} to Q_{tc}. With these matrices, we have

$$\widehat{G}_{tc} = \underbrace{\begin{pmatrix} C_{t_1c_1}E_{t_1c} \\ C_{t_2c_2}E_{t_2c} \end{pmatrix}}_{=:\widehat{V}_{tc}} \widehat{Z}_{tc}^* P_{tc}^*$$

and only have to compute the SVD of $\widehat{V}_{tc}\widehat{Z}_{tc}^*$, choose a rank k_{tc}, and use the first k_{tc} left singular vectors as columns of the matrix $\widehat{Q}_{tc}$, which can be split into

$$\widehat{Q}_{tc} = \begin{pmatrix} F_{t_1c} \\ F_{t_2c} \end{pmatrix}$$

to obtain the transfer matrices for the new cluster basis. In this case, we can use $C_{tc} = \widehat{Q}_{tc}^*\widehat{V}_{tc}$ to compute the basis-change matrix efficiently.

4. Complexity. In order to analyze the complexity of the new algorithms, we follow the approach of [5, section 5]: For the sake of simplicity, we assume that all bounding boxes on the same level are identical up to translation. We also assume that the cluster tree is geometrically regular and that the surface Ω is two-dimensional, i.e., that there are constants $C_{\mathrm{cp}}, C_{\mathrm{nc}}, C_{\mathrm{cu}}, C_{\mathrm{cl}}, C_{\mathrm{ov}}, C_{\mathrm{rs}}, C_{\mathrm{un}} \in \mathbb{R}_{>0}$ such that

$$\begin{aligned}
\operatorname{diam}(B_t) &\le C_{\mathrm{cp}} \operatorname{diam}(B_{t'}) && \text{for all } t \in \mathcal{T}_{\mathcal{I}},\ t' \in \operatorname{chil}(t),\\
\#\operatorname{chil}(t) &\le C_{\mathrm{nc}}, \quad \#\operatorname{chil}(t) \neq 1 && \text{for all } t \in \mathcal{T}_{\mathcal{I}},\\
|\Omega \cap \mathcal{B}(x,r)| &\le C_{\mathrm{cu}} r^2 && \text{for all } x \in \mathbb{R}^3,\ r \in \mathbb{R}_{\ge 0},\\
\operatorname{diam}^2(B_t) &\le C_{\mathrm{cl}} |B_t \cap \Omega| && \text{for all } t \in \mathcal{T}_{\mathcal{I}},\\
\#\{t \in \mathcal{T}_{\mathcal{I}}^{(\ell)} \ :\ x \in B_t\} &\le C_{\mathrm{ov}} && \text{for all } x \in \Omega,\ \ell \in [0 : p_{\mathcal{I}}],\\
C_{\mathrm{lf}}\kappa \operatorname{diam}(B_t) &\le 1 && \text{for all leaves } t \in \mathcal{L}_{\mathcal{I}},\\
C_{\mathrm{rs}}^{-1} k \le \#\hat{t} &\le C_{\mathrm{rs}} k && \text{for all leaves } t \in \mathcal{L}_{\mathcal{I}},\\
\eta_2 \operatorname{dist}(B_t, B_s) < \operatorname{diam}(B_t) &\Rightarrow \#\hat{s} \le C_{\mathrm{un}} \#\hat{t} && \text{for all } t \in \mathcal{L}_{\mathcal{I}},\ s \in \mathcal{T}_{\mathcal{I}}\\
& && \quad \text{with } \operatorname{level}(t) = \operatorname{level}(s).
\end{aligned}$$

The constant C_{cp} ensures that child clusters do not decrease in size too quickly, while C_{nc} provides an upper bound for the number of children. C_{cu} and C_{cl} measure how "convoluted" the surface Ω is, and C_{ov} describes the overlap of clusters. C_{lf} and C_{rs} ensure that the leaves are small enough compared to the wavelength, and C_{un} can be interpreted as a quasi-uniformity condition for neighboring leaf clusters. Additionally, we assume that the number of directions associated with a cluster is bounded, i.e., that there is a constant $C_{\mathrm{di}} \in \mathbb{R}_{>0}$ with

$$\#\mathcal{D}_t \leq C_{\mathrm{di}}(1 + \kappa^2 \operatorname{diam}^2(B_t)) \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}}. \tag{4.1}$$

If the directions are constructed as in Remark 2.3, this condition is satisfied.

According to [5, Lemma 8], there is a *sparsity constant* $C_{\mathrm{sp}} \in \mathbb{R}_{>0}$ such that

$$\sum_{c \in \mathcal{D}_t} \#\operatorname{row}(t,c) \leq \begin{cases} C_{sp} & \text{if } C_{cp}\kappa \operatorname{diam}(B_t) < 1, \\ C_{sp}\kappa^2 \operatorname{diam}(B_t)^2 & \text{otherwise} \end{cases} \tag{4.2}$$

holds for all $t \in \mathcal{T}_{\mathcal{I}}$. We introduce the short notation

$$C_{\mathrm{sp},t} := \begin{cases} C_{\mathrm{sp}} & \text{if } C_{\mathrm{cp}}\kappa \operatorname{diam}(B_t) < 1, \\ C_{\mathrm{sp}}\kappa^2 \operatorname{diam}(B_t)^2 & \text{otherwise} \end{cases} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}}.$$

According to [5, Lemma 9], there is a constant $C_{\mathrm{lv}} \in \mathbb{R}_{>0}$ such that

$$\#\mathcal{T}_{\mathcal{I}}^{(\ell)} \leq C_{\mathrm{lv}} \frac{|\Omega|}{\operatorname{diam}^2(B_t)} \qquad \text{for all } \ell \in [0 : p_{\mathcal{I}}],\ t \in \mathcal{T}_{\mathcal{I}}^{(\ell)}, \tag{4.3a}$$

$$\#\mathcal{T}_{\mathcal{I}} \leq C_{\mathrm{lv}} \frac{\#\mathcal{I}}{k}. \tag{4.3b}$$

The estimates (4.2) and (4.3) give rise to the following fundamental result.

LEMMA 4.1 (block and cluster sums). *There are constants $C_{bs}, C_{cs} \in \mathbb{R}_{>0}$ with*

$$\sum_{t \in \mathcal{T}_{\mathcal{I}}} \sum_{c \in \mathcal{D}_t} \#\operatorname{row}(t,c) \leq C_{bs}\left(\#\mathcal{T}_{\mathcal{I}} + C_{lv}(p_{\mathcal{I}}+1)\kappa^2\right), \tag{4.4a}$$

$$\sum_{t \in \mathcal{T}_{\mathcal{I}}} \#\mathcal{D}_t \leq C_{cs}\left(\#\mathcal{T}_{\mathcal{I}} + C_{lv}(p_{\mathcal{I}}+1)\kappa^2\right). \tag{4.4b}$$

Proof. Combining (4.2) and (4.3a) yields

$$\begin{aligned}
\sum_{t \in \mathcal{T}_{\mathcal{I}}} \sum_{c \in \mathcal{D}_t} \#\operatorname{row}(t,c) &= \sum_{\substack{t \in \mathcal{T}_{\mathcal{I}} \\ C_{\mathrm{cp}}\kappa \operatorname{diam}(B_t) < 1}} \sum_{c \in \mathcal{D}_t} \#\operatorname{row}(t,c) + \sum_{\substack{t \in \mathcal{T}_{\mathcal{I}} \\ C_{\mathrm{cp}}\kappa \operatorname{diam}(B_t) \geq 1}} \sum_{c \in \mathcal{D}_t} \#\operatorname{row}(t,c) \\
&\leq \sum_{\substack{t \in \mathcal{T}_{\mathcal{I}} \\ C_{\mathrm{cp}}\kappa \operatorname{diam}(B_t) < 1}} C_{\mathrm{sp}} + \sum_{\substack{t \in \mathcal{T}_{\mathcal{I}} \\ C_{\mathrm{cp}}\kappa \operatorname{diam}(B_t) \geq 1}} C_{\mathrm{sp}}\kappa^2 \operatorname{diam}^2(B_t) \\
&\leq C_{\mathrm{sp}}\#\mathcal{T}_{\mathcal{I}} + \sum_{\ell=0}^{p_{\mathcal{I}}} \sum_{t \in \mathcal{T}_{\mathcal{I}}^{(\ell)}} C_{\mathrm{sp}}\kappa^2 \operatorname{diam}^2(B_t) \\
&\leq C_{\mathrm{sp}}\#\mathcal{T}_{\mathcal{I}} + \sum_{\ell=0}^{p_{\mathcal{I}}} C_{\mathrm{lv}} \frac{|\Omega|}{\operatorname{diam}^2(B_t)} C_{\mathrm{sp}}\kappa^2 \operatorname{diam}^2(B_t) \\
&\leq C_{\mathrm{sp}}\#\mathcal{T}_{\mathcal{I}} + C_{\mathrm{lv}}C_{\mathrm{sp}}|\Omega|(p_{\mathcal{I}}+1)\kappa^2,
\end{aligned}$$

and we obtain (4.4a) by choosing $C_{\mathrm{bs}} := \max\{C_{\mathrm{sp}}, C_{\mathrm{sp}}|\Omega|\}$.

For the second estimate, we combine (4.1) with (4.3a) to find

$$\begin{aligned}\sum_{t\in\mathcal{T}_\mathcal{I}} \#\mathcal{D}_t &\leq C_{\mathrm{di}} \sum_{t\in\mathcal{T}_\mathcal{I}} (1+\kappa^2 \operatorname{diam}^2(B_t)) = C_{\mathrm{di}} \#\mathcal{T}_\mathcal{I} + C_{\mathrm{di}} \sum_{\ell=0}^{p_\mathcal{I}} \sum_{t\in\mathcal{T}_\mathcal{I}^{(\ell)}} \kappa^2 \operatorname{diam}^2(B_t)\\ &\leq C_{\mathrm{di}} \#\mathcal{T}_\mathcal{I} + C_{\mathrm{di}} \sum_{\ell=0}^{p_\mathcal{I}} C_{\mathrm{lv}} \frac{|\Omega|}{\operatorname{diam}^2(B_t)} \kappa^2 \operatorname{diam}^2(B_t)\\ &= C_{\mathrm{di}} \#\mathcal{T}_\mathcal{I} + C_{\mathrm{di}} (p_\mathcal{I}+1) C_{\mathrm{lv}} |\Omega| \kappa^2,\end{aligned}$$

and we can obtain (4.4b) by choosing $C_{\mathrm{cs}} := \max\{C_{\mathrm{di}}, C_{\mathrm{di}}|\Omega|\}$. ∎

To establish an estimate for the complexity, we need to bound the work of the QR factorization as well as the SVD. We assume that there are constants $C_{\mathrm{qr}}, C_{\mathrm{svd}}$ such that the work of computing the QR factorization and the SVD of a matrix $A \in \mathbb{C}^{m\times n}$ up to machine accuracy is bounded by

$$C_{\mathrm{qr}} mn \min\{m,n\}, \tag{4.5a}$$

$$C_{\mathrm{svd}} mn \min\{m,n\}, \tag{4.5b}$$

respectively.

Now we can consider the complexity of the different phases of the recompression algorithm. We first have to compute the *basis weight matrices* $R_{W,tc}$ for the original cluster basis $(W_{tc})_{t\in\mathcal{T}_\mathcal{I}, c\in\mathcal{D}_t}$.

LEMMA 4.2 (basis weights). *There is a constant $C_{bw} \in \mathbb{R}_{>0}$ such that computing the basis weights $(R_{W,tc})_{t\in\mathcal{T}_\mathcal{I}, c\in\mathcal{D}_t}$ (cf. (3.7)) requires not more than*

$$C_{bw} k^3 \left(\frac{\#\mathcal{I}}{k} + (p_\mathcal{I}+1)\kappa^2 \right) \text{ operations.}$$

Proof. Using [4, Algorithm 16], adapted for multiple directions per cluster, this task takes $(C_{\mathrm{qr}}+2)k^3$ operations per cluster and direction, and Lemma 4.1 together with (4.3b) yields

$$\begin{aligned}\sum_{t\in\mathcal{T}_\mathcal{I}} \sum_{c\in\mathcal{D}_t} (C_{\mathrm{qr}}+2)k^3 &\leq (C_{\mathrm{qr}}+2)k^3 C_{\mathrm{cs}} (\#\mathcal{T}_\mathcal{I} + C_{\mathrm{lv}}(p_\mathcal{I}+1)\kappa^2)\\ &= C_{\mathrm{cs}} C_{\mathrm{lv}} (C_{\mathrm{qr}}+2) k^3 \left(\frac{\#\mathcal{I}}{k} + (p_\mathcal{I}+1)\kappa^2 \right).\end{aligned}$$

We let $C_{\mathrm{bw}} := C_{\mathrm{cs}} C_{\mathrm{lv}} (C_{\mathrm{qr}}+2)$ to complete the proof. ∎

The second step is to compute the *total weight matrices* $\widehat{Z}_{tc}$ for the original cluster basis $(V_{tc})_{t\in\mathcal{T}_\mathcal{I}, c\in\mathcal{D}_t}$.

LEMMA 4.3 (total weights). *There is a constant $C_{we} \in \mathbb{R}_{>0}$ such that computing the total weights $(\widehat{Z}_{tc})_{t\in\mathcal{T}_\mathcal{I}, c\in\mathcal{D}_t}$ requires not more than*

$$C_{we} k^3 \left(\frac{\#\mathcal{I}}{k} + (p_\mathcal{I}+1)\kappa^2 \right) \text{ operations.}$$

Proof. Let $t \in \mathcal{T}_\mathcal{I}$ and $c \in \mathcal{D}_t$.

We have to set up the matrix Z_{tc}. For all $s \in \operatorname{row}(t,c)$, this means computing the product $S_{tsc} R_{W,sc}^*$, which takes not more than $2k^3$ operations.

If there is a corresponding parent cluster t^+, we also have to compute the product of the transfer matrix E_{tc^+} and the parent's weight $\widehat{Z}_{t^+c^+}$ for all $c^+ \in \operatorname{sd}_{t^+}^{-1}(\{c\})$, which takes not more than $2k^3$ operations per product.

We denote the number of columns of Z_{tc} by

$$m := k\#\operatorname{sd}_{t^+}^{-1}(\{c\}) + k\#\operatorname{row}(t,c))$$

and have shown that $2mk^2$ operations are needed to set up this matrix.

Now follows a QR factorization of $Z_{tc}^* \in \mathbb{C}^{m\times k}$, which requires not more than $C_{\mathrm{qr}}mk\min\{m,k\} \leq C_{\mathrm{qr}}mk^2$ operations.

In consequence, the complexity for the whole cluster tree is bounded by

$$\sum_{t\in\mathcal{T}_\mathcal{I}}\sum_{c\in\mathcal{D}_t}(C_{\mathrm{qr}}+2)mk^2 = (C_{\mathrm{qr}}+2)k^3\sum_{t\in\mathcal{T}_\mathcal{I}}\sum_{c\in\mathcal{D}_t}\#\operatorname{sd}_{t^+}^{-1}(\{c\}) + \#\operatorname{row}(t,c).$$

Since sd_{t^+} maps every direction $c^+ \in \mathcal{D}_{t^+}$ to a direction $c = \operatorname{sd}_{t^+}(c^+) \in \mathcal{D}_t$, we have

$$\mathcal{D}_{t^+} = \bigcup_{c\in\mathcal{D}_t}\operatorname{sd}_{t^+}^{-1}(\{c\}), \qquad \#\mathcal{D}_{t^+} = \sum_{c\in\mathcal{D}_t}\#\operatorname{sd}_{t^+}^{-1}(\{c\})$$

and can use Lemma 4.1 (and the convention $\mathcal{D}_{t^+} = \emptyset$ if t is the root) to find the bound

$$\begin{aligned}
\sum_{t\in\mathcal{T}_\mathcal{I}}\sum_{c\in\mathcal{D}_t}(C_{\mathrm{qr}}+2)mk^2 &= (C_{\mathrm{qr}}+2)k^3\sum_{t\in\mathcal{T}_\mathcal{I}}\sum_{c\in\mathcal{D}_t}\#\operatorname{sd}_{t^+}^{-1}(\{c\}) + \#\operatorname{row}(t,c)\\
&= (C_{\mathrm{qr}}+2)k^3\sum_{t\in\mathcal{T}_\mathcal{I}}\#\mathcal{D}_{t^+} + \sum_{c\in\mathcal{D}_t}\#\operatorname{row}(t,c)\\
&= (C_{\mathrm{qr}}+2)k^3\sum_{t^+\in\mathcal{T}_\mathcal{I}}\sum_{t\in\operatorname{chil}(t^+)}\#\mathcal{D}_{t^+}\\
&\quad + (C_{\mathrm{qr}}+2)k^3\sum_{t\in\mathcal{T}_\mathcal{I}}\sum_{c\in\mathcal{D}_t}\#\operatorname{row}(t,c)\\
&\leq (C_{\mathrm{qr}}+2)k^3\sum_{t^+\in\mathcal{T}_\mathcal{I}}C_{\mathrm{nc}}\#\mathcal{D}_{t^+}\\
&\quad + (C_{\mathrm{qr}}+2)k^3\sum_{t\in\mathcal{T}_\mathcal{I}}\sum_{c\in\mathcal{D}_t}\#\operatorname{row}(t,c)\\
&\leq (C_{\mathrm{qr}}+2)C_{\mathrm{nc}}k^3C_{\mathrm{cs}}(\#\mathcal{T}_\mathcal{I} + C_{\mathrm{lv}}(p_\mathcal{I}+1)\kappa^2)\\
&\quad + (C_{\mathrm{qr}}+2)k^3C_{\mathrm{bs}}(\#\mathcal{T}_\mathcal{I} + C_{\mathrm{lv}}(p_\mathcal{I}+1)\kappa^2)\\
&= (C_{\mathrm{qr}}+2)(C_{\mathrm{nc}}C_{\mathrm{cs}}+C_{\mathrm{bs}})k^3(\#\mathcal{T}_\mathcal{I} + C_{\mathrm{lv}}(p_\mathcal{I}+1)\kappa^2).
\end{aligned}$$

We can use (4.3b) to complete the proof with $C_{\mathrm{we}} := (C_{\mathrm{qr}}+2)(C_{\mathrm{nc}}C_{\mathrm{cs}}+C_{\mathrm{bs}})C_{\mathrm{lv}}$. ∎

Now we can address the construction of the improved cluster basis.

LEMMA 4.4 (truncation). *There is a constant $C_{tr} \in \mathbb{R}_{>0}$ such that computing the improved cluster basis $(Q_{tc})_{t\in\mathcal{T}_\mathcal{I},c\in\mathcal{D}_t}$ requires not more than*

$$C_{tr}k^3\left(\frac{\#\mathcal{I}}{k} + (p_\mathcal{I}+1)\kappa^2\right) \text{ operations.}$$

Proof. Let $t \in \mathcal{T}_\mathcal{I}$ and $c \in \mathcal{D}_t$.

If t is a leaf, $V_{tc} \in \mathbb{C}^{m\times k}$, $m := \#\hat{t}$, is used directly. We compute the product $V_{tc}\widehat{Z}_{tc}^* \in \mathbb{C}^{m\times k}$ in not more than $2mk^2$ operations, its SVD in not more than

$C_{\mathrm{svd}} mk \min\{m,k\} \leq C_{\mathrm{svd}} mk^2$ operations, and the basis-change matrix C_{tc} in not more than $2mk^2$ operations.

Due to our assumptions, we have $m = \#\hat{t} \leq C_{\mathrm{rs}} k$, and the number of operations for leaf clusters is bounded by $(C_{\mathrm{svd}} + 4) C_{\mathrm{rs}} k^3$.

If t is not a leaf, we compute the product of the transfer matrix $E_{t'c}$ and the already calculated basis-change matrix $C_{t'c'}$ for every child $t' \in \operatorname{chil}(t)$ and $c' = \operatorname{sd}_{t'}(c)$, and the resulting matrix $\widehat{V}_{tc}$ has $m := \sum_{t' \in \operatorname{chil}(t)} k_{t'c'}$ rows and k columns. Computing all products takes not more than

$$\sum_{t' \in \operatorname{chil}(t)} 2k^2 k_{t'c'} = 2mk^2 \text{ operations.}$$

Now the matrix $V_{tc} \widehat{Z}_{tc}^*$ has to be computed; this takes not more than $2mk^2$ operations. Due to (4.5b), its SVD can be computed in $C_{\mathrm{svd}} mk \min\{m,k\} \leq C_{\mathrm{svd}} mk^2$ operations. Finally, the basis-change matrix C_{tc} can be computed in not more than $2mk^2$ operations.

Due to our assumptions, $\#\operatorname{chil}(t) \leq C_{\mathrm{nc}}$ holds, and we have $m \leq C_{\mathrm{nc}} k$, so the number of operations for nonleaf clusters is bounded by $(C_{\mathrm{svd}} + 6) C_{\mathrm{nc}} k^3$.

Finding the correct ranks k_{tc} requires the inspection of m singular values and can be accomplished in $\mathcal{O}(k)$ operations, so we can conclude that there is a constant C such that not more than Ck^3 operations are required per cluster $t \in \mathcal{T}_{\mathcal{I}}$ and direction $c \in \mathcal{D}_t$.

The total number of operations is bounded by

$$\sum_{t \in \mathcal{T}_{\mathcal{I}}} \sum_{c \in \mathcal{D}_t} Ck^3 = Ck^3 \sum_{t \in \mathcal{T}_{\mathcal{I}}} \#\mathcal{D}_t \leq CC_{\mathrm{cs}} k^3 (\#\mathcal{T}_{\mathcal{I}} + C_{\mathrm{lv}}(p_{\mathcal{I}} + 1)\kappa^2)$$

due to (4.4b), and (4.3b) completes the proof. ∎

The only thing left is the calculation of the new coupling matrices, but this is a simple matrix multiplication of the old coupling matrices with the basis-change matrices (3.8).

LEMMA 4.5 (projections). *There is a constant $C_{pr} \in \mathbb{R}_{>0}$ such that computing the new coupling matrices $(\widetilde{S}_b)_{b \in \mathcal{L}^+_{\mathcal{I} \times \mathcal{I}}}$ requires not more than*

$$C_{pr} k^3 \left(\frac{\#\mathcal{I}}{k} + (p_{\mathcal{I}} + 1)\kappa^2 \right) \text{ operations.}$$

Proof. Computing the products $T_b := C_{tc} S_b$ and $\widetilde{S}_b := T_b C_{sc}^*$ requires not more than $4k^3$ operations for each block $b \in \mathcal{L}^+_{\mathcal{I} \times \mathcal{I}}$. Due to (4.4a), the total number of operations is bounded by

$$\sum_{b \in \mathcal{L}^+_{\mathcal{I} \times \mathcal{I}}} 4k^3 \leq 4k^3 \sum_{t \in \mathcal{T}_{\mathcal{I}}} \sum_{c \in \mathcal{D}_t} \#\operatorname{row}(t,c) \leq 4C_{\mathrm{bs}} k^3 (\#\mathcal{T}_{\mathcal{I}} + C_{\mathrm{lv}}(p_{\mathcal{I}} + 1)\kappa^2),$$

and we can use (4.3b) to obtain our estimate with $C_{\mathrm{pr}} := 4C_{\mathrm{bs}} C_{\mathrm{lv}}$. ∎

For the complete recompression, we have to compute the basis and total weights for the row and the column cluster basis, we have to truncate both bases, and we have to apply the projection to obtain the improved $\mathcal{DH}^2$-matrix representation.

THEOREM 4.6 (complexity). *Let a $\mathcal{DH}^2$-matrix be given. The entire recompression algorithm requires not more than*

$$(2C_{bw} + 2C_{we} + 2C_{tr} + C_{pr})k^3 \left(\frac{\#\mathcal{I}}{k} + (p_{\mathcal{I}} + 1)\kappa^2 \right) \text{ operations.}$$

Proof. The proof follows by simply adding the estimates provided by the previous lemmas. ∎

Remark 4.7 (complexity). Let $n := \#\mathcal{I}$ denote the matrix dimension. If the mesh is quasi-uniform, splitting a cluster will approximately halve the number of indices; therefore, keeping C_{rs} constant requires a tree depth of $p_{\mathcal{I}} \sim \log n$.

If the wave number κ is constant, the first term in the estimate of Theorem 4.6 is dominant, and the recompression algorithm requires $\mathcal{O}(nk^2)$ operations.

In the high-frequency case, we have $\kappa^2 \sim n$, the second term is dominant, and the recompression algorithm requires $\mathcal{O}(nk^3 \log n)$ operations.

5. Numerical experiments. As an example, we use the three-dimensional unit sphere and cube. For the cube, we simply represent each face by two triangles that are then regularly refined. For the sphere, we start with the double pyramid $P = \{x \in \mathbb{R}^3 \ : \ |x_1| + |x_2| + |x_3| = 1\}$, refine every one of its eight faces regularly, and shift the resulting vertices to the unit sphere. For constructing the Galerkin stiffness matrix $G \in \mathbb{C}^{\mathcal{I} \times \mathcal{I}}$, we use piecewise constant basis functions and Sauter–Erichsen–Schwab quadrature of order $n_q = 5$ [23, 10] for triangles that share a vertex or an edge or that are identical and otherwise use Gauss quadrature with Duffy transformation of order $n_q = 3$ [8].

As clustering strategy, the standard binary space partitioning is applied until clusters contain not more than 32 elements. We used $\eta_1 = 10$ for creating the directions (2.1a) and $\eta_2 = 1$ for the standard (2.1b) and parabolic admissibility condition (2.1c). The initial $\mathcal{DH}^2$-matrix approximation is constructed by directional interpolation of order $m = 4$, and the initial rank is $k = 4^3 = 64$.

We choose the wave number in such a way that $\kappa h \approx 0.6$ is ensured; i.e., we have approximately 10 elements per wavelength. For the recompression algorithm, we employ an accuracy $\epsilon = 10^{-4}$ for the blockwise relative Frobenius norm.

We have parallelized the algorithm using the OpenMP standard in order to take advantage of multicore shared-memory systems: Since the basis weights (3.7) are computed by a bottom-up recursion, all weights within one level of the tree can be computed in parallel. Since the total weights (3.6) are computed by a top-down recursion, we can also perform all operations within the same level in parallel. Finally, the construction of the adaptive cluster bases is again a bottom-up procedure and therefore accessible to the same approach. On a server with two Intel Xeon Platinum 8160 processors with a total of 48 cores, direct interpolation for 131 072 triangles takes 203 seconds with the parallel implementation and 5 543 seconds without, a speedup of 27, while recompression takes 1 475 seconds with the parallel version and 38 676 seconds without, a speedup of 26. We have observed that the speedup improves as the problem size grows.

Table 1 shows our results for the single-layer potential on the unit sphere. The first column gives the number n of degrees of freedom, and the second gives the wave number κ. The third and fourth columns show the storage per degree of freedom in KB (1 KB is 1 024 bytes) of the original cluster basis and the original $\mathcal{DH}^2$-matrix, and the fifth, sixth, and seventh columns give the rank, storage for the basis, and storage for the matrix after the recompression. The last two columns give the absolute and

TABLE 1
Single-layer potential operator on the cube (Frobenius norm).

		Original		Recomp.				
n	κ	Basis	Matrix	Rank	Basis	Matrix	Error	Rel. error
2 352	3.5	0.6	51.6	15	0.1	36.1	4.93_{-8}	2.11_{-6}
4 800	5	2.0	166.4	20	0.3	66.6	3.69_{-8}	3.12_{-6}
9 408	7	3.2	326.5	23	0.4	116.5	1.70_{-8}	2.74_{-6}
19 200	10	6.4	496.7	28	0.6	167.9	1.06_{-8}	3.39_{-6}
37 632	14	13.8	941.3	30	1.9	274.8	6.57_{-9}	4.03_{-6}
76 800	20	25.2	1 516.5	33	1.4	374.0	3.80_{-9}	4.64_{-6}
150 528	28	35.3	2 183.7	33	2.0	484.5	2.13_{-9}	4.98_{-6}
307 200	40	57.8	2 730.7	36	3.1	607.1	1.28_{-9}	5.99_{-6}

TABLE 2
Double-layer potential operator on the cube (Frobenius norm).

		Original		Recomp.				
n	κ	Basis	Matrix	Rank	Basis	Matrix	Error	Rel. error
2 352	3.5	0.6	51.6	18	0.1	36.3	1.21_{-7}	2.45_{-7}
4 800	5	2.0	166.4	22	0.3	67.5	1.30_{-7}	3.75_{-7}
9 408	7	3.2	326.5	26	0.4	118.3	8.08_{-8}	3.26_{-7}
19 200	10	6.4	496.7	30	0.8	176.5	1.97_{-7}	1.13_{-6}
37 632	14	13.8	941.3	33	1.5	294.8	2.19_{-7}	1.76_{-6}
76 800	20	25.2	1 516.5	37	2.4	409.5	1.88_{-7}	2.16_{-6}
150 528	28	35.3	2 183.7	37	3.6	541.4	1.46_{-7}	2.36_{-6}
307 200	40	57.8	2 730.7	43	4.4	647.9	6.33_{-8}	1.46_{-6}

relative errors between the $\mathcal{DH}^2$-matrix and its recompressed version, measured in the Frobenius norm.

Similar results are obtained for the double-layer potential; they are presented with the same structure as in Table 2.

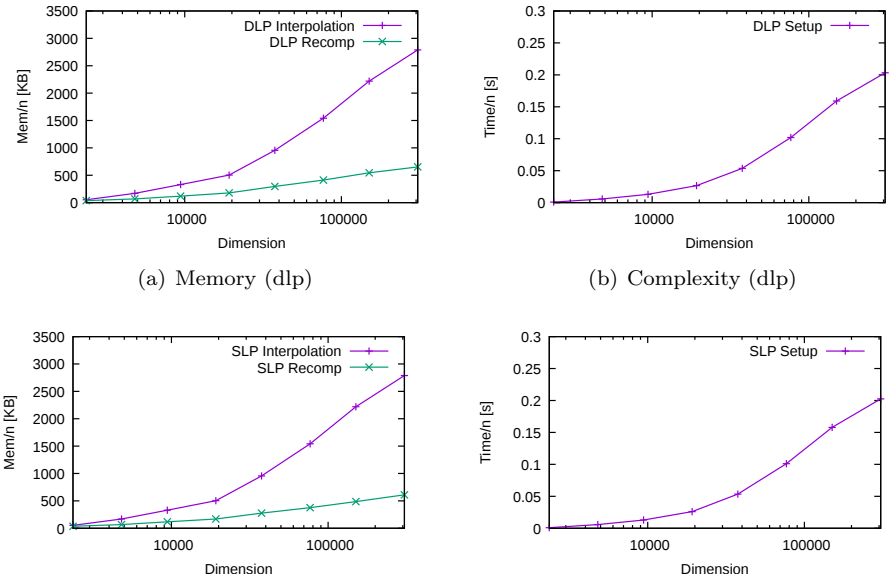
To outline the results, Figure 1 shows the memory requirements per degree of freedom as a function of n for the single-layer (c) and the double-layer potential (a). In both cases, from the beginning of our experiment the recompressed version needs less storage, and the memory advantage improves with n.

Panels (b) and (d) in Figure 1 show the run time per degree of freedom for the recompression of the $\mathcal{DH}^2$-matrix obtained by interpolation. Since we use a logarithmic scale for n and a linear scale for the time divided by n, the experiment confirms our expectation of $\mathcal{O}(n \log n)$ complexity.

Even for higher wave numbers and other norms, the algorithm keeps this behavior: Table 3 shows results for doubled wave numbers on the unit sphere, where the error control strategy for the recompression uses the spectral norm instead of the Frobenius norm.

Next we consider a more realistic geometry: a mesh of an airplane, more precisely a Boeing 747, comprised of 549 836 triangles and 274 920 vertices, provided by courtesy of Boris Dilba. We use the wave number $\kappa = 3.15$, corresponding to a wavelength of approximately 2. The maximal extent of the airplane is approximately 62, i.e., approximately 31 wavelengths. A picture of our object of study is shown in Figure 2.

We have modified our recompression algorithm such that it can be applied during the setup process to reduce intermediate storage requirements: The cluster basis is orthogonalized immediately, and the coupling matrices are constructed on the fly when needed during the recompression algorithm. With the modified algorithm, we are able



(a) Memory (dlp) (b) Complexity (dlp)

(c) Memory (slp) (d) Complexity (slp)

FIG. 1. *Memory and time per degree of freedom for the cube.*

TABLE 3
Single-layer potential operator on the sphere (spectral norm).

		Original		Recomp.				
n	κ	Basis	Matrix	Rank	Basis	Matrix	Error	Rel. error
2 048	8	0.3	35.2	8	0.0	32.3	3.94_{-9}	2.74_{-6}
4 608	12	1.3	82.9	9	0.1	71.3	3.22_{-9}	6.65_{-6}
8 192	16	4.9	205.5	12	0.2	120.6	1.63_{-9}	7.37_{-6}
18 432	24	10.1	821.9	15	0.6	223.0	1.26_{-9}	1.66_{-5}
32 768	32	45.7	1 450.8	15	1.7	312.3	6.15_{-10}	1.74_{-5}
73 728	48	72.7	3 336.2	18	2.8	459.3	2.62_{-10}	2.16_{-5}
131 072	64	98.4	5 114.2	21	3.9	586.7	1.49_{-10}	2.64_{-5}

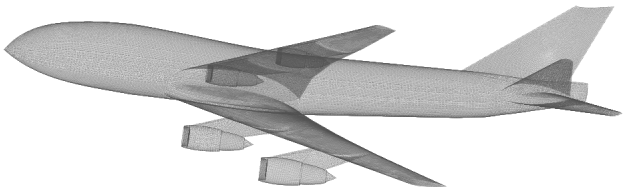


FIG. 2. *Mesh of a Boeing* 747.

to set up the $\mathcal{DH}^2$-matrix with linear basis functions for the airplane mesh to obtain the results given in Table 4, where we have varied both the recompression tolerance ϵ and the interpolation order m. We also report run times (in hours) measured on our shared-memory system.

We have measured the relative error between the dense, i.e., uncompressed, matrix and the recompressed approximation in the Frobenius norm. To put these results in perspective, the dense matrix takes about 4 296 KB per degree of freedom.

TABLE 4
Boeing 747 with single-layer potential (direct recompression).

ϵ	m	Time	Rank	Basis	Matrix	Error	Rel. error
1.0_{-2}	4	0.5	28	1.7	83.6	1.0_{-3}	4.9_{-2}
1.0_{-3}	5	1.3	38	2.9	105.9	1.8_{-4}	8.4_{-4}
1.0_{-4}	6	3.2	46	4.7	138.4	3.1_{-5}	1.5_{-4}

Compared to interpolation, our recompression algorithm reduces the storage requirements for the cluster basis from 194 to 1.7 KB for order $m = 4$ and from 2 137 to 4.7 KB for order $m = 6$. For the coupling matrices, we save similar amounts of storage: We go from 2 322 to 83.6 KB for order $m = 4$ and from 25 833 to 138.4 KB for order $m = 6$. The measured relative Frobenius error is always well below the prescribed tolerance.

We can see that $\mathcal{DH}^2$-recompression is absolutely crucial in order to turn the initial approximation constructed by directional interpolation into a practically useful representation that saves approximately 96% of storage at an accuracy of 3.1×10^{-5}.

REFERENCES

[1] S. AMINI AND A. T. J. PROFIT, *Analysis of a diagonal form of the fast multipole algorithm for scattering theory*, BIT, 39 (1999), pp. 585–602.

[2] L. BANJAI AND W. HACKBUSCH, *$\mathcal{H}$- and $\mathcal{H}^2$-matrices for low and high frequency Helmholtz equations*, IMA J. Numer. Anal., 28 (2008), pp. 46–79.

[3] M. BEBENDORF, C. KUSKE, AND R. VENN, *Wideband nested cross approximation for Helmholtz problems*, Numer. Math., 130 (2015), pp. 1–34.

[4] S. BÖRM, *Efficient Numerical Methods for Non-Local Operators: $\mathcal{H}^2$-Matrix Compression, Algorithms and Analysis*, EMS Tracts Math. 14, European Mathematical Society, Zürich, Switzerland, 2010.

[5] S. BÖRM, *Directional $\mathcal{H}^2$-matrix compression for high-frequency problems*, Numer. Linear. Alg. Appl., 24 (2017), https://doi.org/10.1002/nla.2112.

[6] S. BÖRM AND J. M. MELENK, *Approximation of the high-frequency Helmholtz kernel by nested directional interpolation: Error analysis*, Numer. Math., 137 (2017), pp. 1–34, https://doi.org/10.1007/s00211-017-0873-y.

[7] A. BRANDT, *Multilevel computations of integral transforms and particle interactions with oscillatory kernels*, Comput. Phys. Commun., 65 (1991), pp. 24–38.

[8] M. G. DUFFY, *Quadrature over a pyramid or cube of integrands with a singularity at a vertex*, SIAM J. Numer. Anal., 19 (1982), pp. 1260–1262.

[9] B. ENGQUIST AND L. YING, *Fast directional multilevel algorithms for oscillatory kernels*, SIAM J. Sci. Comput., 29 (2007), pp. 1710–1737.

[10] S. ERICHSEN AND S. A. SAUTER, *Efficient automatic quadrature in 3-d Galerkin BEM*, Comput. Methods Appl. Mech. Engrg., 157 (1998), pp. 215–224.

[11] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Johns Hopkins University Press, London, 1996.

[12] L. GRASEDYCK AND W. HACKBUSCH, *Construction and arithmetics of $\mathcal{H}$-matrices*, Computing, 70 (2003), pp. 295–334.

[13] L. GREENGARD, J. HUANG, V. ROKHLIN, AND S. WANDZURA, *Accelerating fast multipole methods for the Helmholtz equation at low frequencies*, IEEE Comp. Sci. Eng., 5 (1998), pp. 32–38.

[14] L. GREENGARD AND V. ROKHLIN, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.

[15] W. HACKBUSCH, *A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part* I: *Introduction to $\mathcal{H}$-matrices*, Computing, 62 (1999), pp. 89–108.

[16] W. HACKBUSCH AND B. N. KHOROMSKIJ, *A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part* II: *Application to multi-dimensional problems*, Computing, 64 (2000), pp. 21–47.

[17] W. HACKBUSCH AND Z. P. NOWAK, *On the fast matrix multiplication in the boundary element method by panel clustering*, Numer. Math., 54 (1989), pp. 463–491.

[18] M. Messner, M. Schanz, and E. Darve, *Fast directional multilevel summation for oscillatory kernels based on Chebyshev interpolation*, J. Comput. Phys., 231 (2012), pp. 1175–1196.
[19] E. Michielssen and A. Boag, *A multilevel matrix decomposition algorithm for analyzing scattering from large structures*, IEEE Trans. Antennas and Propagation, AP-44 (1996), pp. 1086–1093.
[20] V. Rokhlin, *Rapid solution of integral equations of classical potential theory*, J. Comput. Phys., 60 (1985), pp. 187–207.
[21] V. Rokhlin, *Diagonal forms of translation operators for the Helmholtz equation in three dimensions*, Appl. Comput. Harmon. Anal., 1 (1993), pp. 82–93.
[22] S. A. Sauter, *Variable order panel clustering*, Computing, 64 (2000), pp. 223–261.
[23] S. A. Sauter and C. Schwab, *Boundary Element Methods*, Springer, New York, 2011.