Max-Planck-Institut
für Mathematik
in den Naturwissenschaften
Leipzig

Construction of data-sparse $\mathcal{H}^2$-matrices by hierarchical compression

(revised version: April 2008)

by

Steffen Börm

Preprint no.: 92 2007

CONSTRUCTION OF DATA-SPARSE $\mathcal{H}^2$-MATRICES BY HIERARCHICAL COMPRESSION

STEFFEN BÖRM*

Abstract. Hierarchical matrices ($\mathcal{H}$-matrices) provide an elegant approach to handling large densely populated matrices: the matrix is split into a hierarchy of blocks, and each block is approximated by a low-rank matrix in factorized form. It has been demonstrated that this representation can be used to treat integral and partial differential equations, solve matrix equations from the field of control theory, and evaluate matrix functions efficiently.

$\mathcal{H}^2$-matrices use a refined representation that employs a multi-level structure in order to reduce the storage requirements of hierarchical matrices. It has been shown that $\mathcal{H}^2$-matrices can significantly reduce storage requirements for large problems, in particular when combined with modern error control schemes.

Until now, all algorithms for constructing an efficient approximation of a general matrix by an $\mathcal{H}^2$-matrix required a representation of the *entire* original matrix to be kept in storage, therefore the storage requirements of $\mathcal{H}^2$-matrix algorithms could be far larger than those of the final approximation. This paper presents a new approach that allows us to construct an $\mathcal{H}^2$-matrix without storing the entire original matrix. The central idea is to approximate submatrices and combine them by an efficient new algorithm to form approximations of larger matrices until the entire matrix has been treated. Using this new approach, many $\mathcal{H}$-matrix algorithms can be "refitted" easily to compute results in the more efficient representation.

Possible applications include efficient matrix arithmetics for the construction of preconditioners, the approximation of matrix functions or solutions of matrix equations, or efficient compression schemes based on the popular cross approximation algorithms.

Key words. Hierarchical matrices, data-sparse approximation, non-local operators

AMS subject classifications. 65F30, 65N38

Acknowledgement. A significant part of this work was carried out during a stay at the Institut für Geometrie und Praktische Mathematik of the RWTH Aachen.

1. Introduction.

The performance of a numerical scheme depends crucially on the representation of input and output data, e.g., finite element methods for partial differential equations are only efficient because they make use of the sparsity of the stiffness matrix, the same holds for solvers like the conjugate gradient methods or multigrid schemes.

There are many problems that lead to non-sparse matrices: most integral equations have non-local kernel functions, therefore standard discretizations lead to non-sparse stiffness matrices. The LU or Cholesky factorizations of sparse matrices are in general also not sparse, and the same holds for solutions of certain matrix equations or the results of matrix functions.

Hierarchical matrices (abbreviated $\mathcal{H}$-matrices) [20, 21, 14, 17, 9] provide a flexible method for treating these non-sparse matrices: following the general idea of panel-clustering [23, 25] and multipole [24, 19] techniques, the matrix is split into a disjoint partition of submatrices, and each submatrix is approximated by a low-rank matrix in factorized form.

It is possible to prove that integral equations [8], solution operators of elliptic partial differential equations [2, 6], solutions of matrix equations [16, 18] and matrix functions [12, 13] can be approximated efficiently by $\mathcal{H}$-matrices.

*Max-Planck-Institut für Mathematik in den Naturwissenschaften, Inselstraße 22–26, 04103 Leipzig, Germany (sbo@mis.mpg.de)

$\mathcal{H}$-matrix algorithms have by now matured to a point that allows even very large problems to be treated in a relatively short time, provided that enough storage is available. Since the storage capacity of typical computers grows at a much slower rate than their processing power, we have to look for techniques that use the available capacity as efficiently as possible.

$\mathcal{H}^2$-matrices [22, 10] replace the general low-rank representation of the matrix blocks by a special multi-level representation that takes advantage of the relationships between different submatrices in order to reduce the storage requirements. This representation was originally only available for certain integral operators, but by now algorithms have been developed that can approximate arbitrary matrices up to an arbitrary error tolerance [10, 7, 5]. It has been proven that this representation is efficient for integral operators [5] and solution operators of elliptic partial differential equations [6].

The algorithms mentioned above construct an $\mathcal{H}^2$-matrix approximation of the given matrix in one pass and require the entire original matrix to be available during the computation. This is a severe limitation: the algorithm may fail if the original matrix does not fit into the storage capacity even though there would be sufficient capacity for the compressed $\mathcal{H}^2$-matrix.

This paper introduces a new approach: only submatrices of the original matrix are constructed, the submatrices are compressed to get $\mathcal{H}^2$-submatrices, and these compressed matrices are then recursively merged to form the entire $\mathcal{H}^2$-matrix. Since only compressed matrices are stored, the overall storage requirements no longer depend on the original matrix, thus the new algorithm can treat significantly larger problems than its predecessors.

A second advantage of the new technique is that it fits the structure of most $\mathcal{H}$-matrix algorithms, therefore these algorithms can be "refitted" to compute results in the more efficient $\mathcal{H}^2$-matrix representation instead of the $\mathcal{H}$-matrix representation.

The paper is organized in four sections: section 1 is the introduction you are currently reading, section 2 defines the basic structure of $\mathcal{H}^2$-matrices, section 3 recalls the fundamental theory of $\mathcal{H}^2$-matrix compression and error control and presents the new hierarchical compression scheme, and section 4 contains numerical experiments that demonstrate its efficiency.

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

We will now briefly recall the structure of $\mathcal{H}^2$-matrices [22, 10].

2.1. Block structure.

Hierarchical matrix techniques are based on detecting subblocks of the matrix which admit a data-sparse approximation. In order to find these *admissible* blocks efficiently, we introduce a hierarchy of subsets:

DEFINITION 2.1 (Cluster tree). *Let $\mathcal{I}$ be an index set. Let $\mathcal{T}$ be a labeled tree. We denote its root by* $\operatorname{root}(\mathcal{T})$*, the label of $t \in \mathcal{T}$ by $\hat{t}$, and the set of sons by* $\operatorname{sons}(\mathcal{T}, t)$ *(or just* $\operatorname{sons}(t)$ *if this does not lead to ambiguity).*

$\mathcal{T}$ is a cluster tree *for $\mathcal{I}$ if it satisfies the following conditions:*

- $\widehat{\operatorname{root}(\mathcal{T})} = \mathcal{I}$.
- *If* $\operatorname{sons}(t) \neq \emptyset$ *holds for $t \in \mathcal{T}$, we have*

$$\hat{t} = \bigcup_{s \in \operatorname{sons}(t)} \hat{s} \quad \text{and}$$

$$\hat{s}_1 \cap \hat{s}_2 = \emptyset \text{ for all } s_1, s_2 \in \operatorname{sons}(t) \text{ with } s_1 \neq s_2.$$

If $\mathcal{T}$ is a cluster tree for $\mathcal{I}$, we will denote it by $\mathcal{T}_\mathcal{I}$ and call its nodes clusters. *The*

set of leaves *of* $\mathcal{T}_\mathcal{I}$ *is denoted by*

$$\mathcal{L}_\mathcal{I} := \{t \in \mathcal{T}_\mathcal{I} \ : \ \operatorname{sons}(t) = \emptyset\}.$$

The definition implies $\hat{t} \subseteq \mathcal{I}$ for all clusters $t \in \mathcal{T}_\mathcal{I}$. We can use induction to prove that the set $\mathcal{L}_\mathcal{I}$ of leaves of $\mathcal{T}_\mathcal{I}$ is a disjoint partition of the index set $\mathcal{I}$, i.e.,

$$\mathcal{I} = \dot{\bigcup}_{t \in \mathcal{L}_\mathcal{I}} \hat{t}. \tag{2.1}$$

Given a cluster tree $\mathcal{T}_\mathcal{I}$ and a cluster $t \in \mathcal{T}_\mathcal{I}$, we denote the subtree with root t by $\mathcal{T}_\mathcal{I}^t$ and call its nodes, including t itself, the *descendants* of t. If $t \in \mathcal{T}_\mathcal{I}$ is a descendant of $t^+ \in \mathcal{T}_\mathcal{I}$, the cluster t^+ is called a *predecessor* of t. The *set of predecessors* of a cluster $t \in \mathcal{T}_\mathcal{I}$ is defined by

$$\operatorname{pred}(t) := \{t^+ \in \mathcal{T}_\mathcal{I} \ : \ t \in \mathcal{T}_\mathcal{I}^{t^+}\}.$$

Using cluster trees, we can now define a hierarchical partition of the matrix entries:

DEFINITION 2.2 (Block cluster tree). *Let $\mathcal{I}$, $\mathcal{J}$ be index sets, and let $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ be corresponding cluster trees. Let $\mathcal{T}$ be a labeled tree. $\mathcal{T}$ is a* block cluster tree *for $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ if it satisfies the following conditions:*

- $\operatorname{root}(\mathcal{T}) = (\operatorname{root}(\mathcal{T}_\mathcal{I}), \operatorname{root}(\mathcal{T}_\mathcal{J}))$.
- *Each cluster $b \in \mathcal{T}$ has the form $b = (t, s)$ for $t \in \mathcal{T}_\mathcal{I}$ and $s \in \mathcal{T}_\mathcal{J}$ and its label satisfies $\hat{b} = \hat{t} \times \hat{s}$.*
- *Let $b = (t, s) \in \mathcal{T}$. If $\operatorname{sons}(b) \neq \emptyset$, we have*

$$\operatorname{sons}(b) = \begin{cases} \{t\} \times \operatorname{sons}(s) & \text{if } \operatorname{sons}(t) = \emptyset, \operatorname{sons}(s) \neq \emptyset, \\ \operatorname{sons}(t) \times \{s\} & \text{if } \operatorname{sons}(t) \neq \emptyset, \operatorname{sons}(s) = \emptyset \\ \operatorname{sons}(t) \times \operatorname{sons}(s) & \text{otherwise.} \end{cases}$$

If $\mathcal{T}$ is a block cluster tree for $\mathcal{I}$ and $\mathcal{J}$, we will denote it by $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ and call its nodes blocks. *The leaves of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ are denoted by $\mathcal{L}_{\mathcal{I}\times\mathcal{J}}$.*

This definition implies that a block cluster tree for $\mathcal{I}$ and $\mathcal{J}$ is a cluster tree for the product index set $\mathcal{I} \times \mathcal{J}$, therefore the set of leaf labels

$$P := \{\hat{b} = \hat{t} \times \hat{s} \ : \ b = (t, s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}\}$$

defines a disjoint partition of $\mathcal{I} \times \mathcal{J}$ into blocks of indices.

DEFINITION 2.3 (Admissibility). *Let $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ be cluster trees for $\mathcal{I}$ and $\mathcal{J}$. Let $\alpha : \mathcal{T}_\mathcal{I} \times \mathcal{T}_\mathcal{J} \to \mathbb{B} = \{true, false\}$ be a predicate, which we will call* admissibility condition. *A block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is called* admissible *if*

$$(\operatorname{sons}(\mathcal{T}_\mathcal{I}, t) \neq \emptyset \vee \operatorname{sons}(\mathcal{T}_\mathcal{J}, s) \neq \emptyset) \Longrightarrow \alpha(t, s) \qquad \text{holds for all } b = (t, s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}},$$

i.e., if each leaf of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is either admissible or a pair of leaf clusters of the respective cluster trees. For an admissible block cluster tree, we split the set of leaves into

$$\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}} := \{b = (t, s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}} \ : \ \alpha(t, s) \text{ holds}\} \quad \text{and} \quad \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}} := \mathcal{L}_{\mathcal{I}\times\mathcal{J}} \setminus \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}},$$

i.e., into admissible and inadmissible leaves. Usually, we will not work with α, but only use $\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ and $\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}$.

For matrices resulting from the discretization of elliptic problems, the admissibility condition

$$\alpha(t,s) := \begin{cases} \text{true} & \text{if } \max\{\operatorname{diam}(\Omega_t), \operatorname{diam}(\Omega_s)\} \leq \operatorname{dist}(\Omega_t, \Omega_s), \\ \text{false} & \text{otherwise} \end{cases} \tag{2.2}$$

is frequently used, where Ω_t and Ω_s are suitable domains containing the supports of the basis functions or functionals corresponding to t and s.

The condition (2.2) ensures that we are dealing with a region where we can expect Green's function to be smooth or at least separable. In the case $\mathcal{I} = \mathcal{J}$, this means that the block $\hat{t} \times \hat{s}$ lies "sufficiently far away" from the diagonal of the matrix.

If the indices in $\mathcal{I}$ and $\mathcal{J}$ correspond to locations in space, it is possible to construct good cluster trees $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ by binary space partitioning and a good block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ by a simple recursion strategy [14, 17].

2.2. Matrix structure. Typical hierarchical matrices are defined based on the leaves $\mathcal{L}_{\mathcal{I}\times\mathcal{J}}$ of a block cluster tree: the submatrices $M|_{\hat{t}\times\hat{s}}$ corresponding to admissible leaf blocks $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ are required to be of low rank:

$$\mathcal{H}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, k) := \{M \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}} \ : \ \operatorname{rank}(M|_{\hat{t}\times\hat{s}}) \leq k \text{ for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}\}.$$

The parameter $k \in \mathbb{N}$ is called the *local rank* of the $\mathcal{H}$-matrix set. For each $b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$, the low-rank matrix $M|_{\hat{t}\times\hat{s}}$ is represented in factorized form, i.e., by matrices $A_b \in \mathbb{R}^{\hat{t}\times k}$ and $B_b \in \mathbb{R}^{\hat{s}\times k}$ with $M|_{\hat{t}\times\hat{s}} = A_b B_b^\top$.

The $\mathcal{H}^2$-matrix format is a specialization of this representation: we require not only that admissible blocks correspond to low-rank submatrices, but also that the ranges of these blocks and their adjoints are contained in predefined spaces.

Definition 2.4 (Cluster basis). *Let $\mathcal{T}_\mathcal{I}$ be a cluster tree, and let $K = (K_t)_{t\in\mathcal{T}_\mathcal{I}}$ be a family of index sets. A family $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ of matrices satisfying $V_t \in \mathbb{R}^{\hat{t}\times K_t}$ for all $t \in \mathcal{T}_\mathcal{I}$ is called* cluster basis *for $\mathcal{T}_\mathcal{I}$ and K. For each $t \in \mathcal{T}_\mathcal{I}$, the cardinality $\#K_t$ is called the* rank *of V in t.*

The high efficiency of $\mathcal{H}^2$-matrix methods is owed to the fact that the cluster bases are designed to form a nested hierarchy matching the cluster tree:

Definition 2.5 (Nested cluster basis). *Let $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ be a cluster basis for a cluster tree $\mathcal{T}_\mathcal{I}$ and a family $(K_t)_{t\in\mathcal{T}_\mathcal{I}}$ of index sets. Let $E = (E_t)_{t\in\mathcal{T}_\mathcal{I}}$ be a family of matrices satisfying $E_{t'} \in \mathbb{R}^{K_{t'}\times K_t}$ and*

$$(V_t)|_{\hat{t}'\times K_t} = V_{t'} E_{t'} \qquad \text{for all } t \in \mathcal{T}_\mathcal{I}, t' \in \operatorname{sons}(t). \tag{2.3}$$

Then the cluster basis V is called nested *with* transfer matrices E.

If a cluster basis $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ is nested, we have

$$V_t = \begin{pmatrix} V_{t_1} E_{t_1} \\ \vdots \\ V_{t_\tau} E_{t_\tau} \end{pmatrix} = \begin{pmatrix} V_{t_1} & & \\ & \ddots & \\ & & V_{t_\tau} \end{pmatrix} \begin{pmatrix} E_{t_1} \\ \vdots \\ E_{t_\tau} \end{pmatrix}$$

for all $t \in \mathcal{T}_\mathcal{I}$ with $\#\operatorname{sons}(t) = \tau > 0$ and $\operatorname{sons}(t) = \{t_1, \ldots, t_\tau\}$. This means that we have to store the matrices V_t only for leaf clusters $t \in \mathcal{L}_\mathcal{I}$ and can use the transfer matrices $E_{t'}$ to represent them *implicitly* for all other clusters. Since the transfer matrices $E_{t'}$ only require $(\#K_{t'})(\#K_t)$ units of storage, while the cluster basis matrices V_t require $(\#\hat{t})(\#K_t)$ units, this nested representation is very efficient for $\#\hat{t} \gg \#K_{t'}$.

The nested structure is the key difference between general hierarchical matrices and $\mathcal{H}^2$-matrices [22, 10], since it allows us to construct very efficient algorithms by re-using information across the entire cluster tree, similar to multigrid algorithms or other multilevel methods.

DEFINITION 2.6 ($\mathcal{H}^2$-matrix). *Let $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ be cluster trees. Let $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ be an admissible block cluster tree. Let V and W be nested cluster bases for $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ with families $K = (K_t)_{t\in\mathcal{T}_\mathcal{I}}$ and $L = (L_s)_{s\in\mathcal{T}_\mathcal{J}}$ of index sets. The set of $\mathcal{H}^2$-matrices for $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, V and W is given by*

$$\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W) := \{X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}} \ : \text{for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}} \text{ there is a matrix } S_b \in \mathbb{R}^{K_t\times L_s} \text{ satisfying } X|_{\hat{t}\times\hat{s}} = V_t S_b W_s^\top\}$$

In this context, V is called the row cluster basis*, W is called the* column cluster basis*, and the family $S = (S_b)_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}$ is called the family of* coupling matrices*.*

2.3. Complexity. Let us now consider the storage complexity of the $\mathcal{H}^2$-matrix representation.

Block cluster trees constructed for standard situations have an important property: for each $t \in \mathcal{T}_\mathcal{I}$, there is only a limited number of blocks of the form (t, s), i.e., the cardinalities of the sets

$$\operatorname{row}(t) := \{s \in \mathcal{T}_\mathcal{J} \ : \ (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}\}, \qquad \operatorname{col}(s) := \{t \in \mathcal{T}_\mathcal{I} \ : \ (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}\}$$

can be bounded by a constant. For cluster trees and block cluster trees constructed by geometric bisection, an explicit bound can be given, and this bound does not depend on the number of degrees of freedom [14, 17].

DEFINITION 2.7 (Sparsity). *Let $C_{\mathrm{sp}} \in \mathbb{N}$. The block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is* C_{sp}-sparse *if we have*

$$\#\operatorname{row}(t) = \#\{s \in \mathcal{T}_\mathcal{J} \ : \ (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}\} \leq C_{\mathrm{sp}} \qquad \text{for all } t \in \mathcal{T}_\mathcal{I}, \tag{2.4a}$$

$$\#\operatorname{col}(s) = \#\{t \in \mathcal{T}_\mathcal{I} \ : \ (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}\} \leq C_{\mathrm{sp}} \qquad \text{for all } s \in \mathcal{T}_\mathcal{J}. \tag{2.4b}$$

The complexity of an $\mathcal{H}^2$-matrix representation can be bounded if the following conditions are fulfilled:

- the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is admissible and C_{sp}-sparse,
- each cluster has only a bounded number of sons, i.e., there is a constant $C_{\mathrm{sn}} \geq 1$ with

$$\#\operatorname{sons}(t) \leq C_{\mathrm{sn}}, \qquad \#\operatorname{sons}(s) \leq C_{\mathrm{sn}} \qquad \text{for all } t \in \mathcal{T}_\mathcal{I}, s \in \mathcal{T}_\mathcal{J}, \tag{2.5}$$

- each leaf cluster is not too large, i.e., there is a constant $C_{\mathrm{lf}} \geq 1$ satisfying

$$\#\hat{t} \leq C_{\mathrm{lf}} \# K_t, \qquad \#\hat{s} \leq C_{\mathrm{lf}} \# L_s \qquad \text{for all leaves } t \in \mathcal{L}_\mathcal{I}, s \in \mathcal{L}_\mathcal{J}. \tag{2.6}$$

To keep the notations short, we introduce the abbreviations

$$k_t := \#K_t, \qquad l_s := \#L_s \qquad \text{for all } t \in \mathcal{T}_\mathcal{I}, s \in \mathcal{T}_\mathcal{J}.$$

Let us first consider the storage complexity of the cluster bases V and W.

LEMMA 2.8 (Storage of a cluster basis). *A nested cluster basis $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$, represented using transfer matrices, requires not more than*

$$(C_{\mathrm{lf}} + C_{\mathrm{sn}}^{1/2}) \sum_{t\in\mathcal{T}_\mathcal{I}} k_t^2 \quad \textit{units of storage.}$$

Proof. For each leaf cluster $t \in \mathcal{T}_\mathcal{I}$, we have to store the matrix V_t, which requires $(\#\hat{t})(\#K_t)$ units of storage. Due to (2.6), this is bounded by $C_{\mathrm{lf}}(\#K_t)^2 \leq C_{\mathrm{lf}} k_t^2$, and *all* of these matrices require not more than

$$C_{\mathrm{lf}} \sum_{t \in \mathcal{L}_\mathcal{I}} k_t^2 \leq C_{\mathrm{lf}} \sum_{t \in \mathcal{T}_\mathcal{I}} k_t^2 \quad \text{units of storage.}$$

For each non-leaf cluster $t \in \mathcal{T}_\mathcal{I}$, we have to store the transfer matrices $E_{t'}$ for all $t' \in \operatorname{sons}(t)$, which requires $(\#K_{t'})(\#K_t) = k_{t'} k_t$ units of storage and a total of

$$\begin{aligned}\sum_{t \in \mathcal{T}_\mathcal{I}} \sum_{t' \in \operatorname{sons}(t)} k_t k_{t'} &\leq \left(\sum_{t \in \mathcal{T}_\mathcal{I}} \sum_{t' \in \operatorname{sons}(t)} k_t^2 \right)^{1/2} \left(\sum_{t \in \mathcal{T}_\mathcal{I}} \sum_{t' \in \operatorname{sons}(t)} k_{t'}^2 \right)^{1/2} \\ &\overset{(2.5)}{\leq} \left(C_{\mathrm{sn}} \sum_{t \in \mathcal{T}_\mathcal{I}} k_t^2 \right)^{1/2} \left(\sum_{t' \in \mathcal{T}_\mathcal{I}} k_{t'}^2 \right)^{1/2} = C_{\mathrm{sn}}^{1/2} \sum_{t \in \mathcal{T}_\mathcal{I}} k_t^2,\end{aligned}$$

where we have used the Cauchy-Schwarz inequality and the fact that each cluster has at most one father. □

The storage requirements of the coupling matrices and the nearfield part of an $\mathcal{H}^2$-matrix can be bounded using the sparsity of $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$:

LEMMA 2.9 (Storage of matrices). *The coupling matrices $(S_b)_{b \in \mathcal{L}^+_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}}}}$ and the nearfield matrices $(X|_{\hat{b}})_{b \in \mathcal{L}^-_{\mathcal{T}_{\mathcal{I}\times\mathcal{J}}}}$ for an $\mathcal{H}^2$-matrix X require not more than*

$$\frac{C_{\mathrm{lf}}^2 C_{\mathrm{sp}}}{2} \left(\sum_{t \in \mathcal{T}_\mathcal{I}} k_t^2 + \sum_{s \in \mathcal{T}_\mathcal{J}} l_s^2 \right) \quad \textit{units of storage.}$$

Proof. Similar to Lemma 2.8. □

Adding the estimates of Lemma 2.8 and Lemma 2.9 yields an upper bound of

$$\left(\frac{C_{\mathrm{lf}}^2 C_{\mathrm{sp}}}{2} + C_{\mathrm{lf}} + C_{\mathrm{sn}}^{1/2} \right) \left(\sum_{t \in \mathcal{T}_\mathcal{I}} k_t^2 + \sum_{s \in \mathcal{T}_\mathcal{J}} l_s^2 \right) \tag{2.7}$$

for the storage requirements of an $\mathcal{H}^2$-matrix.

In simple situations, we can assume that the ranks of the matrices V_t and W_s are bounded by a constant $k \in \mathbb{N}$, i.e., that

$$\#K_t \leq k, \qquad \#L_s \leq k \qquad \text{holds for all } t \in \mathcal{T}_\mathcal{I}, s \in \mathcal{T}_\mathcal{J}.$$

This leads to an estimate of $\mathcal{O}((\#\mathcal{T}_\mathcal{I} + \#\mathcal{T}_\mathcal{J})k^2)$ for the storage requirements.

In practical applications, the cluster trees are constructed in such a way that $\#\mathcal{T}_\mathcal{I} \lesssim n/k$ and $\#\mathcal{T}_\mathcal{J} \lesssim n/k$ hold for

$$n := \max\{\#\mathcal{I}, \#\mathcal{J}\},$$

therefore an $\mathcal{H}^2$-matrix representation of X requires only $\mathcal{O}(nk)$ units of storage.

In the general case, we can take advantage of the fact that the estimate (2.7) depends only on the sums of k_t^2 and l_s^2 and not on the maximum k, therefore we can admit higher ranks for a small number of clusters and still get a low complexity [25, 5].

3. Construction of cluster bases. Our goal is to approximate an arbitrary matrix $X \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ by an $\mathcal{H}^2$-matrix $\widetilde{X}$. We assume that the cluster trees $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ and the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ are given.

3.1. Matrix approximation. If the row and column cluster bases V and W are also given, it is reasonable to wonder if we can compute the best approximation of X in the matrix space $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$.

DEFINITION 3.1 (Orthogonal cluster basis). *Let $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ be a cluster basis. It is called* orthogonal *if*

$$V_t^\top V_t = I \qquad \text{holds for all } t \in \mathcal{T}_\mathcal{I}. \tag{3.1}$$

If V and W are orthogonal cluster bases, the computation of the least-squares approximation of X in $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$ is straightforward: for an admissible block $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$, we let $X_{t,s} := X|_{\hat{t}\times\hat{s}}$ and look for a good approximation of this block in the factorized form $V_t S_b W_s^\top$ used by $\mathcal{H}^2$-matrices. We let $S_b^* := V_t^\top X_{t,s} W_s$ and observe that the orthogonality of V_t and W_s implies

$$\begin{aligned}
\|X_{t,s} - V_t\widetilde{S}_b W_s^\top\|_F^2 &= \|X_{t,s} - V_t S_b^* W_s + V_t(S_b^* - \widetilde{S}_b)W_s^\top\|_F^2 \\
&= \|X_{t,s} - V_t S_b^* W_s\|_F^2 + \|V_t(S_b^* - \widetilde{S}_b)W_s^\top\|_F^2 \\
&= \|X_{t,s} - V_t S_b^* W_s\|_F^2 + \|S_b^* - \widetilde{S}_b\|_F^2 \quad \text{for all } \widetilde{S}_b \in \mathbb{R}^{K_t\times L_s},
\end{aligned}$$

i.e., S_b^* minimizes the blockwise approximation error. For the Frobenius norm, the sum of the squares of the blockwise approximation errors yields the square of the total approximation error, therefore the $\mathcal{H}^2$-matrix $\widetilde{X}$ defined by $\widetilde{X}|_{\hat{t}\times\hat{s}} = V_t S_b^* W_s^\top$ for all $b \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ is indeed the best approximation of X in the space $\mathcal{H}^2(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, V, W)$ with respect to the Frobenius norm. Using

$$\begin{aligned}
\|X_{t,s} - V_t S_b^* W_s^\top\|_F^2 &= \|X_{t,s} - V_t V_t^\top X_{t,s} + V_t V_t^\top X_{t,s} - V_t V_t^\top X_{t,s} W_s W_s^\top\|_F^2 \\
&= \|X_{t,s} - V_t V_t^\top X_{t,s}\|_F^2 + \|V_t V_t^\top (X_{t,s} - X_{t,s} W_s W_s^\top)\|_F^2 \\
&\leq \|X_{t,s} - V_t V_t^\top X_{t,s}\|_F^2 + \|X_{t,s} - X_{t,s} W_s W_s^\top\|_F^2,
\end{aligned} \tag{3.2}$$

the influence of V and W can be investigated independently.

We conclude that finding the matrix $\widetilde{X}$ is a relatively simple task once we have good orthogonal cluster bases at our disposal, and that orthogonal row and column cluster bases V and W can be considered "good" if

$$\|X_{t,s} - V_t V_t^\top X_{t,s}\|_F \leq \epsilon, \quad \|X_{t,s}^\top - W_s W_s^\top X_{t,s}^\top\|_F \leq \epsilon \quad \text{for all } b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$$

hold for a given error tolerance $\epsilon \in \mathbb{R}_{>0}$.

Due to the symmetry of these inequalities, we can restrict our attention to the construction of a good row cluster basis V, since applying the same technique to the transposed matrix $X^\top$ will then yield a good column cluster basis W.

In practical applications, the Frobenius norm is usually only of little interest, more important are induced matrix norms like the spectral norm. The theory presented here carries over to this case [5].

3.2. Orthogonalization. Let us now consider the construction of "good" cluster bases. In a first step, we assume that a nested cluster basis $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ for the cluster tree $\mathcal{T}_\mathcal{I}$ and the index sets $K = (K_t)_{t\in\mathcal{T}_\mathcal{I}}$ is given that satisfies

$$\min_{Z_b\in\mathbb{R}^{K_t\times\hat{s}}} \|X_{t,s} - V_t Z_b\|_F \leq \epsilon \qquad \text{for all } b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}. \tag{3.3}$$

We are looking for an *orthogonal* nested cluster basis $Q = (Q_t)_{t\in\mathcal{T}_\mathcal{I}}$ for the same cluster tree and the new index sets $(\widetilde{K}_t)_{t\in\mathcal{T}_\mathcal{I}}$ that satisfies

$$\|X_{t,s} - Q_t Q_t^\top X_{t,s}\|_F \leq \epsilon \qquad \text{for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}},$$

i.e., that allows us to compute the best approximation of $X_{t,s}$ explicitly by an orthogonal projection.

In general, the latter inequality will only hold if the range of V_t is contained in the range of Q_t, i.e., if there is a matrix $R_t \in \mathbb{R}^{\widetilde{K}_t\times K_t}$ such that $V_t = Q_t R_t$ holds.

This equation already suggests a good approach for an algorithm: we are looking for orthogonal factorizations of the matrices V_t, and we have the additional requirement that the new cluster basis $Q = (Q_t)_{t\in\mathcal{T}_\mathcal{I}}$ has to be nested. To illustrate the idea, let us consider a cluster $t \in \mathcal{T}_\mathcal{I}$ with $\operatorname{sons}(t) = \{t_1, t_2\}$ for $t_1 \neq t_2$. Since V is nested, there are transfer matrices E_{t_1} and E_{t_2} such that

$$V_t = \begin{pmatrix} V_{t_1} E_{t_1} \\ V_{t_2} E_{t_2} \end{pmatrix}$$

holds. We assume that the factorizations $V_{t_1} = Q_{t_1} R_{t_1}$ and $V_{t_2} = Q_{t_2} R_{t_2}$ and the corresponding index sets $\widetilde{K}_{t_1}$ and $\widetilde{K}_{t_2}$ have already been computed. We also assume that the $\widetilde{K}_{t_1} \subseteq \hat{t}_1$ and $\widetilde{K}_{t_2} \subseteq \hat{t}_2$ hold, since this implies $\widetilde{K}_{t_1} \cap \widetilde{K}_{t_2} = \emptyset$ and allows us to avoid formal problems in the construction of the matrix $\widehat{V}_t$ below. We get

$$V_t = \begin{pmatrix} V_{t_1} E_{t_1} \\ V_{t_2} E_{t_2} \end{pmatrix} = \begin{pmatrix} Q_{t_1} R_{t_1} E_{t_1} \\ Q_{t_2} R_{t_2} E_{t_2} \end{pmatrix} = \begin{pmatrix} Q_{t_1} & \\ & Q_{t_2} \end{pmatrix} \begin{pmatrix} R_{t_1} E_{t_1} \\ R_{t_2} E_{t_2} \end{pmatrix} = \begin{pmatrix} Q_{t_1} & \\ & Q_{t_2} \end{pmatrix} \widehat{V}_t$$

for the auxiliary matrix

$$\widehat{V}_t := \begin{pmatrix} R_{t_1} E_{t_1} \\ R_{t_2} E_{t_2} \end{pmatrix} \in \mathbb{R}^{\widehat{K}_t \times K_t}, \qquad \widehat{K}_t := \widetilde{K}_{t_1} \dot\cup \widetilde{K}_{t_2}.$$

We compute the factorization

$$\widehat{V}_t = \widehat{Q}_t R_t$$

with an orthogonal matrix $\widehat{Q}_t \in \mathbb{R}^{\widehat{K}_t \times \widetilde{K}_t}$ and a matrix $R_t \in \mathbb{R}^{\widetilde{K}_t \times K_t}$ using Givens or Householder transformations. The index set $\widetilde{K}_t$ is chosen as a subset of $\widehat{K}_t \subseteq \hat{t}$ of cardinality $\min\{\#\widehat{K}_t, \#K_t\}$. The new cluster basis matrix Q_t is defined by

$$Q_t := \begin{pmatrix} Q_{t_1} & \\ & Q_{t_2} \end{pmatrix} \widehat{Q}_t,$$

and it is obviously orthogonal. Due to $\widehat{K}_t = \widetilde{K}_{t_1} \dot\cup \widetilde{K}_{t_2}$, we can split $\widehat{Q}_t$ into its "upper half" $F_{t_1} := \widehat{Q}_t|_{\widetilde{K}_{t_1}\times\widetilde{K}_t}$ and its "lower half" $F_{t_2} := \widehat{Q}_t|_{\widetilde{K}_{t_2}\times\widetilde{K}_t}$ and observe

$$\widehat{Q}_t = \begin{pmatrix} F_{t_1} \\ F_{t_2} \end{pmatrix}, \qquad Q_t = \begin{pmatrix} Q_{t_1} & \\ & Q_{t_2} \end{pmatrix} \widehat{Q}_t = \begin{pmatrix} Q_{t_1} & \\ & Q_{t_2} \end{pmatrix} \begin{pmatrix} F_{t_1} \\ F_{t_2} \end{pmatrix} = \begin{pmatrix} Q_{t_1} F_{t_1} \\ Q_{t_2} F_{t_2} \end{pmatrix},$$

i.e., the cluster basis $Q = (Q_t)_{t\in\mathcal{T}_\mathcal{I}}$ is nested (cf. Definition 2.5), and the transfer matrices are given by F_t. Due to our construction, we have

$$V_t = \begin{pmatrix} Q_{t_1} & \\ & Q_{t_2} \end{pmatrix} \widehat{V}_t = \begin{pmatrix} Q_{t_1} & \\ & Q_{t_2} \end{pmatrix} \widehat{Q}_t R_t = Q_t R_t,$$

i.e., the matrices Q_t and R_t indeed form an orthogonal decomposition of V_t.

If t is a leaf, the situation is simple: we let $\widehat{K}_t := \hat{t}$, $\widehat{V}_t := V_t$ and $Q_t := \widehat{Q}_t$, i.e., we just compute a standard orthogonal factorization of V_t by Householder or Givens transformations.

Algorithm 1 Given $X \in \mathbb{R}^{M\times N}$, construct an index set $K \subseteq M$ of minimal cardinality and orthogonal $Q \in \mathbb{R}^{M\times K}$ such that $\|X - QQ^\top X\|_F \leq \epsilon$

procedure Lowrank(X, ϵ, **var** Q, K);
$m \leftarrow \#M$; $\{\mu_1, \ldots, \mu_m\} \leftarrow M$; $n \leftarrow \#N$; $\{\nu_1, \ldots, \nu_n\} \leftarrow N$;
$\widehat{X} \leftarrow 0 \in \mathbb{R}^{m\times n}$;
for $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, n\}$ **do** $\widehat{X}_{ij} \leftarrow X_{\mu_i\nu_j}$;
Compute singular value decomposition $\widehat{X} = U\Sigma V^\top$;
$k \leftarrow \min\{m, n\}$; $\tilde{\epsilon} \leftarrow 0$;
while $k > 0$ and $\tilde{\epsilon} + \Sigma_{kk}^2 \leq \epsilon^2$ **do begin** $k \leftarrow k - 1$; $\tilde{\epsilon} \leftarrow \tilde{\epsilon} + \Sigma_{kk}^2$ **end**;
$K \leftarrow \{\mu_1, \ldots, \mu_k\}$; $Q \leftarrow 0 \in \mathbb{R}^{M\times K}$;
for $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, k\}$ **do** $Q_{\mu_i\mu_j} \leftarrow U_{ij}$

3.3. Truncation. The complexity of subsequent computations with the new cluster basis Q is determined by the cardinalities of the index sets $\widetilde{K} = (\widetilde{K}_t)_{t\in\mathcal{T}_\mathcal{I}}$, therefore we would like these sets to be as small as possible. The orthogonalization procedure guarantees $\#\widetilde{K}_t \leq \#K_t$ for all $t \in \mathcal{T}_\mathcal{I}$, i.e., the new cluster basis will at least not be less efficient than the original one, but this is not necessarily the optimal result.

In order to improve efficiency, we replace the exact factorization $V_t = Q_t R_t$ by an *approximate* factorization: we are looking for an orthogonal nested cluster basis $Q = (Q_t)_{t\in\mathcal{T}_\mathcal{I}}$ such that $V_t \approx Q_t R_t$ holds for the optimal coefficient matrices $R_t := Q_t^\top V_t$. We aim to choose the cardinalities $\#\widetilde{K}_t$ as small as possible given the desired accuracy of the approximation.

This goal is reached by the *truncation* algorithm: we replace the exact factorization $\widehat{V}_t = \widehat{Q}_t R_t$ used in the orthogonalization algorithm by an approximate factorization $\widehat{V}_t \approx \widehat{Q}_t R_t$, e.g., by computing the singular value decomposition of $\widehat{V}_t$ and dropping all singular values below a given error tolerance (cf. Algorithm 1).

In this setting we have

$$V_t = \begin{pmatrix} V_{t_1}E_{t_1} \\ V_{t_2}E_{t_2} \end{pmatrix} \approx \begin{pmatrix} Q_{t_1}R_{t_1}E_{t_1} \\ Q_{t_2}R_{t_2}E_{t_2} \end{pmatrix} = \begin{pmatrix} Q_{t_1} & \\ & Q_{t_2} \end{pmatrix} \widehat{V}_t \approx \begin{pmatrix} Q_{t_1} & \\ & Q_{t_2} \end{pmatrix} \widehat{Q}_t R_t = Q_t R_t,$$

i.e., the approximation error $V_t - Q_t R_t$ is the sum of two errors: the first one is caused by the approximation of V_{t_1} and V_{t_2}, the second one is caused by the approximation of $\widehat{V}_t$. Using the orthogonality of Q_{t_1} and Q_{t_2} yields

$$\begin{aligned} \|(V_t - Q_t R_t)x\|_2^2 &= \left\| \begin{pmatrix} (V_{t_1} - Q_{t_1}R_{t_1})E_{t_1}x \\ (V_{t_2} - Q_{t_2}R_{t_2})E_{t_2}x \end{pmatrix} + \begin{pmatrix} Q_{t_1} & \\ & Q_{t_2} \end{pmatrix} (\widehat{V}_t - \widehat{Q}_t R_t)x \right\|_2^2 \\ &= \|(V_{t_1} - Q_{t_1}R_{t_1})E_{t_1}x\|_2^2 + \|(V_{t_2} - Q_{t_2}R_{t_2})E_{t_2}x\|_2^2 + \|(\widehat{V}_t - \widehat{Q}_t R_t)x\|_2^2 \end{aligned}$$

for all vectors $x \in \mathbb{R}^{K_t}$. The first two terms of this sum are of the same structure as the original term, only with the vectors $E_{t_1}x$ and $E_{t_2}x$ instead of x. Using the *long-range transfer matrices* defined by

$$E_{r,t} := \begin{cases} E_{r,t'}E_{t'} & \text{if } r \in \mathcal{T}_\mathcal{I}^{t'} \text{ for a } t' \in \operatorname{sons}(t), \\ I & \text{otherwise, i.e., if } r = t \end{cases} \qquad \text{for } t \in \mathcal{T}_\mathcal{I}, r \in \mathcal{T}_\mathcal{I}^t,$$

we can proceed by induction (cf. [4, Theorem 4] for a detailed general proof) to get

$$\|(V_t - Q_t R_t)x\|_2^2 = \sum_{r \in \mathcal{T}_{\mathcal{I}}^t} \|(\widehat{V}_r - \widehat{Q}_r R_r) E_{r,t} x\|_2^2. \tag{3.4}$$

This equation relates the total error to the local errors introduced in each step of the algorithm, therefore it can be used to implement sophisticated error-control schemes. Applying it to canonical unit vectors and summing up yields

$$\|V_t - Q_t R_t\|_F^2 = \sum_{r \in \mathcal{T}_{\mathcal{I}}^t} \|(\widehat{V}_r - \widehat{Q}_r R_r) E_{r,t}\|_F^2, \tag{3.5}$$

while taking the supremum of (3.4) yields a similar estimate for the spectral norm.

3.4. Matrix compression. Let us now return our attention to the problem of finding a "good" nested cluster basis for an arbitrary matrix $X \in \mathbb{R}^{\mathcal{I} \times \mathcal{J}}$.

Let $b = (t, s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}^+$ be an admissible block. According to (3.2), we have to find an orthogonal nested cluster basis $Q = (Q_t)_{t \in \mathcal{T}_{\mathcal{I}}}$ such that

$$\|X_{t,s} - Q_t Q_t^\top X_{t,s}\|_F \leq \epsilon$$

holds for a given error tolerance $\epsilon \in \mathbb{R}_{>0}$.

We approach this problem by using the concept of the *total cluster basis* introduced in [7]: if we collect the relevant matrix blocks $X_{t,s}$ in a large matrix X_t with

$$X_{t,s} = X_t|_{\hat{t}\times\hat{s}} \tag{3.6}$$

and find an algorithm that guarantees

$$\|X_t - Q_t Q_t^\top X_t\|_F \leq \epsilon,$$

this would immediately imply

$$\|X_{t,s} - Q_t Q_t^\top X_{t,s}\|_F = \|(X_t - Q_t Q_t^\top X_t)|_{\hat{t}\times\hat{s}}\|_F \leq \|X_t - Q_t Q_t^\top X_t\|_F \leq \epsilon.$$

We choose the matrices X_t in such a way that they not only satisfy (3.6), but also form a nested cluster basis: we let

$$X_t := X|_{\hat{t}\times N_t}, \qquad N_t := \dot{\bigcup} \{\hat{s} \;:\; \text{with } (t^+, s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}^+ \text{ for } t^+ \in \operatorname{pred}(t), s \in \mathcal{T}_{\mathcal{J}}\}$$

for all clusters $t \in \mathcal{T}_{\mathcal{I}}$. Obviously $(X_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ is a cluster basis with the index sets $(N_t)_{t\in\mathcal{T}_{\mathcal{I}}}$, and since $N_{t'} \supseteq N_t$ holds for all $t \in \mathcal{T}_{\mathcal{I}}$, $t' \in \operatorname{sons}(t)$, this cluster basis is also nested with trivial transfer matrices. For all $b = (t, s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}^+$, we have $\hat{s} \subseteq N_t$ and therefore $X_{t,s} = X_t|_{\hat{t}\times\hat{s}}$.

This means that the *total cluster basis* $(X_t)_{t\in\mathcal{T}_{\mathcal{I}}}$ can be used to represent each admissible block of X with zero error. The ranks of the total cluster basis are usually too large (on the order of $\#\mathcal{J}$) for useful algorithms, but we already know how to fix this: we apply the truncation algorithm to the total cluster basis. Due to the simplicity of its transfer matrices, the error estimate (3.5) takes the form

$$\|X_t - Q_t R_t\|_F^2 = \sum_{r \in \mathcal{T}_{\mathcal{I}}^t} \|(\widehat{X}_r - \widehat{Q}_r R_r)|_{\widehat{K}_r \times N_t}\|_F^2, \tag{3.7}$$

and due to $\hat{s} \subseteq N_t$ and $X_t|_{\hat{t}\times\hat{s}} = X_{t,s}$ for all admissible leaves $(t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$, we conclude

$$\|X_{t,s} - Q_t Q_t^\top X_{t,s}\|_F^2 = \sum_{r\in\mathcal{T}_\mathcal{I}^t} \|\widehat{X}_{r,s} - \widehat{Q}_r \widehat{Q}_r^\top \widehat{X}_{r,s}\|_F^2$$

for $\widehat{X}_{r,s} := \widehat{X}_r|_{\widehat{K}_r\times\hat{s}}$, therefore we can control the approximation error by making sure that the local errors on the right-hand side of this equation are under control.

3.5. Unification. Using the total cluster basis $(X_t)_{t\in\mathcal{T}_\mathcal{I}}$ directly to construct the adaptive cluster basis $(Q_t)_{t\in\mathcal{T}_\mathcal{I}}$ is only an option if nothing about the structure of X is known, since it means working with ranks on the order of $\#\mathcal{J}$ and leads to algorithms of at least quadratic complexity.

We consider the special matrix

$$X = \begin{pmatrix} X_1 & \dots & X_p \end{pmatrix}$$

for submatrices $X_i \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}_i}$ with disjoint index sets $\mathcal{J}_1, \dots, \mathcal{J}_p$. We assume that cluster trees $\mathcal{T}_{\mathcal{J}_i}$ are given for each $i \in \{1,\dots,p\}$ and that each of the matrices X_i is an $\mathcal{H}^2$-matrix with nested cluster bases $V_i = (V_{i,t})_{t\in\mathcal{T}_\mathcal{I}}$ and $W_i = (W_{i,s})_{s\in\mathcal{T}_{\mathcal{J}_i}}$ and coupling matrices $(S_{i,b})_{b\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}_i}}$.

In order to make the computation of $Q = (Q_t)_{t\in\mathcal{T}_\mathcal{I}}$ as efficient as possible, we have to take the special structure of the $\mathcal{H}^2$-matrices X_i into account.

Let $t \in \mathcal{T}_\mathcal{I}$, $t^+ \in \operatorname{pred}(t)$ and $s \in \mathcal{T}_{\mathcal{J}_i}$ with $(t^+,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}_i}$. Since X_i is an $\mathcal{H}^2$-matrix, we have

$$X_{i,t^+,s} := (X_i)|_{\hat{t}^+\times\hat{s}} = V_{i,t^+} S_{i,b} W_{i,s}^\top,$$

and since V_i is a nested cluster basis, we can apply (2.3) inductively to get

$$V_{i,t^+}|_{\hat{t}\times K_{i,t^+}} = V_{i,t} E_{i,t,t^+}$$

and therefore

$$X_i|_{\hat{t}\times\hat{s}} = (V_{i,t^+} S_{i,b} W_{i,s}^\top)|_{\hat{t}\times\hat{s}} = V_{i,t^+}|_{\hat{t}\times K_{i,t^+}} S_{i,b} W_{i,s}^\top = V_{i,t} E_{i,t,t^+} S_{i,b} W_{i,s}^\top.$$

This means that the total cluster basis $(X_{i,t})_{t\in\mathcal{T}_\mathcal{I}}$ for the submatrix X_i satisfies

$$X_{i,t} = V_{i,t} Z_{i,t}^\top \qquad \text{for all } t \in \mathcal{T}_\mathcal{I},$$

where $Z_{i,t} \in \mathbb{R}^{N_{i,t}\times K_{i,t}}$ is given by

$$Z_{i,t}|_{\hat{s}\times K_{i,t}} = W_{i,s} S_{i,b}^\top E_{i,t,t^+}^\top \qquad \text{for all } t \in \mathcal{T}_\mathcal{I}, t^+ \in \operatorname{pred}(t), s \in \mathcal{T}_{\mathcal{J}_i} \text{ with } b = (t^+,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}_i}.$$

The matrix $Z_{i,t}$ has only $\#K_{i,t}$ columns, therefore we can use an orthogonal transformation to turn it into a upper triangular matrix with not more than $\#K_{i,t}$ columns. More precisely, we can find an index set $\widetilde{N}_{i,t} \subseteq N_{i,t}$ with $\#\widetilde{N}_{i,t} \leq \#K_{i,t}$, an orthogonal matrix $P_{i,t} \in \mathbb{R}^{N_{i,t}\times\widetilde{N}_{i,t}}$ and a matrix $\widetilde{Z}_{i,t} \in \mathbb{R}^{\widetilde{N}_{i,t}\times K_{i,t}}$ with

$$Z_{i,t} = P_{i,t} \widetilde{Z}_{i,t} \qquad \text{for all } t \in \mathcal{T}_\mathcal{I}.$$

The *weight matrix* $\widetilde{Z}_{i,t}$ is relatively small, since both the number of rows and the number of columns are bounded by $\#K_{i,t}$, and using the factorization

$$X_{i,t} = V_{i,t}\widetilde{Z}_{i,t}^\top P_{i,t}^\top \tag{3.8}$$

allows us to handle the truncation algorithm far more efficiently: the orthogonality of the matrix $P_{i,t}$ implies

$$\|X_{i,t} - Q_t Q_t^\top X_{i,t}\|_F = \|V_{i,t}\widetilde{Z}_{i,t}^\top - Q_t Q_t^\top V_{i,t}\widetilde{Z}_{i,t}^\top\|_F, \tag{3.9a}$$

$$\|\widehat{X}_{i,t} - \widehat{Q}_t\widehat{Q}_t^\top \widehat{X}_{i,t}\|_F = \|\widehat{V}_{i,t}\widetilde{Z}_{i,t}^\top - \widehat{Q}_t\widehat{Q}_t^\top \widehat{V}_{i,t}\widetilde{Z}_{i,t}^\top\|_F \tag{3.9b}$$

for the matrices $R_{i,t} := Q_t^\top V_{i,t}$ and

$$\widehat{V}_{i,t} = \begin{cases} V_{i,t} & \text{if } \operatorname{sons}(t) = \emptyset, \\ \begin{pmatrix} R_{i,t_1} E_{i,t_1} \\ R_{i,t_2} E_{i,t_2} \end{pmatrix} & \text{otherwise} \end{cases}$$

already used in the orthogonalization algorithm.

Replacing all matrices X_t^i in the truncation algorithm by the more compact matrices $V_t^i \widetilde{Z}_t^i$ means that we can handle the matrix

$$\widetilde{X}_t := \begin{pmatrix} V_{1,t}\widetilde{Z}_{1,t}^\top & \dots & V_{p,t}\widetilde{Z}_{p,t}^\top \end{pmatrix} \in \mathbb{R}^{\hat{t}\times\widehat{N}_t}, \qquad \widehat{N}_t := \widetilde{N}_{1,t} \cup \ldots \cup \widetilde{N}_{p,t} \subseteq N_t \tag{3.10}$$

instead of the full total cluster basis matrix X_t without changing the result of the algorithm. We will call the matrix $\widetilde{X}_t$ the *condensed* counterpart of X_t: as far as our algorithm is concerned, both matrices contain essentially the same information, but $\widetilde{X}_t$ is far smaller than X_t. Using this approach, the matrices $\widehat{X}_t$ appearing in the orthogonalization and truncation procedures are given by

$$\widehat{X}_t := \begin{pmatrix} \widehat{V}_{1,t}\widetilde{Z}_{1,t}^\top & \dots & \widehat{V}_{p,t}\widetilde{Z}_{p,t}^\top \end{pmatrix} \in \mathbb{R}^{\widehat{K}_t\times\widehat{N}_t}.$$

We can proceed as in the truncation algorithm: the singular value decomposition of $\widehat{X}_t$ yields an orthogonal matrix $\widehat{Q}_t$ with minimal rank satisfying an error estimate of the form

$$\|\widehat{X}_t - \widehat{Q}_t\widehat{Q}_t^\top \widehat{X}_t\|_F \leq \epsilon_t$$

for an arbitrary $\epsilon_t \in \mathbb{R}_{>0}$. Due to (3.9) and (3.7), we get

$$\|X_{t,s} - Q_t Q_t^\top X_{t,s}\|_F^2 \leq \|X_t - Q_t Q_t^\top X_t\|_F^2 \leq \sum_{r\in\mathcal{T}_\mathcal{I}^t} \epsilon_t^2 \qquad \text{for all } b = (t,s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}^+$$

and conclude that by choosing the error tolerances $(\epsilon_t)_{t\in\mathcal{T}_\mathcal{I}}$ small enough an arbitrarily good approximation can be computed. A similar result [7, Theorem 4.2] can be derived for the total approximation error.

During the course of the algorithm, we also need the operators $R_{i,t} = Q_t^\top V_{i,t}$ describing the mapping from the old cluster bases to the new one. A direct computation would be too time-consuming, but since the orthogonality of Q_{t_1} and Q_{t_2} implies

$$R_{i,t} = Q_t^\top V_{i,t} = \widehat{Q}_t^\top \widehat{V}_{i,t},$$

Algorithm 2 Given $X \in \mathbb{R}^{M\times N}$, construct an index set $K \subseteq M$ and a factorization $X = QR$ such that $Q \in \mathbb{R}^{M\times K}$ is orthogonal, $R \in \mathbb{R}^{K\times N}$ and $\#K \leq \#N$

procedure Householder(X, **var** Q, R, K);
$m \leftarrow \#M$; $\{\mu_1, \ldots, \mu_m\} \leftarrow M$; $n \leftarrow \#N$; $\{\nu_1, \ldots, \nu_n\} \leftarrow N$; $k \leftarrow \min\{m, n\}$;
$\widehat{X} \leftarrow 0 \in \mathbb{R}^{m\times n}$; $\widehat{Q} \leftarrow I \in \mathbb{R}^{m\times m}$;
for $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, n\}$ **do** $\widehat{X}_{ij} \leftarrow X_{\mu_i\nu_j}$;
Apply k Householder reflections to $\widehat{X}$ to make it upper triangular;
Apply the same reflections to $\widehat{Q}$;
$K \leftarrow \{\mu_1, \ldots, \mu_k\}$; $Q \leftarrow 0 \in \mathbb{R}^{M\times K}$; $R \leftarrow 0 \in \mathbb{R}^{K\times N}$
for $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, k\}$ **do** $Q_{\mu_i\mu_j} \leftarrow \widehat{Q}_{ji}$;
for $i \in \{1, \ldots, k\}$, $j \in \{1, \ldots, n\}$ **do** $R_{\mu_i\nu_j} \leftarrow \widehat{X}_{ij}$

we can perform this task more efficiently.

We intend to use this basis construction recursively, so we require the weight matrices $(\widetilde{Z}_t)_{t\in\mathcal{T}_\mathcal{I}}$ for the unified approximation with the new cluster basis Q. Fortunately, their construction is straightforward: the total cluster basis matrices for the unified approximation are given by $Q_t Q_t^\top X_t$, and due to (3.8) and (3.10), we have

$$\begin{aligned} Q_t Q_t^\top X_t &= Q_t Q_t^\top \begin{pmatrix} X_{1,t} & \ldots & X_{p,t} \end{pmatrix} = Q_t Q_t^\top \begin{pmatrix} V_{1,t}\widetilde{Z}_{1,t}^\top P_{1,t}^\top & \ldots & V_{p,t}\widetilde{Z}_{p,t}^\top P_{p,t}^\top \end{pmatrix} \\ &= Q_t Q_t^\top \widetilde{X}_t \begin{pmatrix} P_{1,t} & & \\ & \ddots & \\ & & P_{p,t} \end{pmatrix}^\top = Q_t \widetilde{Y}_t \begin{pmatrix} P_{1,t} & & \\ & \ddots & \\ & & P_{p,t} \end{pmatrix}^\top \end{aligned}$$

for the matrix $\widetilde{Y}_t := Q_t^\top \widetilde{X}_t \in \mathbb{R}^{\widetilde{K}_t\times\widehat{N}_t}$, and we can use a standard orthogonal factorization (cf. Algorithm 2) to find an index set $\widetilde{N}_t \subseteq \widehat{N}_t$, a matrix $\widetilde{Z}_t \in \mathbb{R}^{\widetilde{N}_t\times\widetilde{K}_t}$ and an orthogonal matrix $\widehat{P}_t \in \mathbb{R}^{\widehat{N}_t\times\widetilde{N}_t}$ with

$$\widetilde{Y}_t^\top = \widehat{P}_t \widetilde{Z}_t,$$

therefore we get

$$Q_t Q_t^\top X_t = Q_t \widetilde{Y}_t \begin{pmatrix} P_{1,t} & & \\ & \ddots & \\ & & P_{p,t} \end{pmatrix}^\top = Q_t \widetilde{Z}_t^\top P_t^\top$$

for the orthogonal matrix

$$P_t := \begin{pmatrix} P_{1,t} & & \\ & \ddots & \\ & & P_{p,t} \end{pmatrix} \widehat{P}_t,$$

and conclude that $\widetilde{Z}_t$ is a weight matrix for the new approximation defined by the projection into the new cluster basis Q. The resulting recursive procedure is given in Algorithm 3.

Algorithm 3 Construct unified cluster basis $V = (V_t)_{t\in\mathcal{T}_\mathcal{I}}$ with weight matrices $(\widetilde{Z}_t)_{t\in\mathcal{T}_\mathcal{I}}$ for cluster bases $V_1,\ldots,V_p$ with weight matrices $\widetilde{Z}_1,\ldots,\widetilde{Z}_p$

procedure Unify(t, V_1, …, V_p, $\widetilde{Z}_1$, …, $\widetilde{Z}_p$, **var** Q, $\widetilde{Z}$, R_1, …, R_p);
if $\operatorname{sons}(t) = \emptyset$ **then begin**
  $\widehat{K}_t \leftarrow \hat{t}$;
  for $i \in \{1,\ldots,p\}$ **do** $\widehat{V}_{i,t} \leftarrow V_{i,t}$
end else begin
  $\widehat{K}_t \leftarrow \emptyset$;
  for $t' \in \operatorname{sons}(t)$ **do begin**
    Unify(t', V_1, …, V_p, $\widetilde{Z}_1$, …, $\widetilde{Z}_p$, Q, $\widetilde{Z}$, R_1, …, R_p);
    $\widehat{K}_t \leftarrow \widehat{K}_t \dot\cup \widetilde{K}_{t'}$
  end;
  for $i \in \{1,\ldots,p\}$ **do begin**
    $\widehat{V}_{i,t} \leftarrow 0 \in \mathbb{R}^{\widehat{K}_t \times K_{i,t}}$;
    for $t' \in \operatorname{sons}(t)$ **do** $\widehat{V}_{i,t}|_{\widetilde{K}_{t'}\times K_{i,t}} \leftarrow R_{i,t'} E_{i,t'}$
  end
end;
$\widehat{N}_t \leftarrow \emptyset$;
for $i \in \{1,\ldots,p\}$ **do** $\widehat{N}_t \leftarrow \widehat{N}_t \dot\cup \widetilde{N}_{i,t}$;
$\widehat{X}_t \leftarrow 0 \in \mathbb{R}^{\widehat{K}_t \times \widehat{N}_t}$;
for $i \in \{1,\ldots,p\}$ **do** $\widehat{X}_t|_{\widehat{K}_t\times\widetilde{N}_{i,t}} \leftarrow \widehat{V}_{i,t}\widetilde{Z}_{i,t}$;
Lowrank($\widehat{X}_t$, ϵ_t, $\widehat{Q}_t$, $\widetilde{K}_t$);
for $i \in \{1,\ldots,p\}$ **do** $R_{i,t} \leftarrow \widehat{Q}_t^\top \widehat{V}_{i,t}$;
$\widetilde{Y}_t \leftarrow \widehat{Q}_t^\top \widehat{X}_t \in \mathbb{R}^{\widetilde{K}_t \times \widehat{N}_t}$;
Householder($\widetilde{Y}_t^\top$, $\widehat{P}_t$, $\widetilde{Z}_t$, $\widetilde{N}_t$);
if $\operatorname{sons}(t) = \emptyset$ **then**
  $Q_t \leftarrow \widehat{Q}_t$
else
  for $t' \in \operatorname{sons}(t)$ **do** $F_{t'} \leftarrow \widehat{Q}_t|_{\widetilde{K}_{t'}\times\widetilde{K}_t}$

3.6. Complexity of the unification. Let us now investigate the complexity of Algorithm 3. We assume that the computation of the Householder factorization of a $m \times n$ matrix requires $C_{\mathrm{qr}}mn^2$ operations and that its singular value decomposition (up to machine accuracy) can be found in $C_{\mathrm{svd}}mn^2$ operations.

We introduce the abbreviation

$$k_{i,t} := \#K_{i,t}, \qquad \text{for all } t \in \mathcal{T}_\mathcal{I}$$

and start by establishing a number of basic estimates: by construction, we have

$$\#\widetilde{N}_{i,t} \leq \#K_{i,t} = k_{i,t} \qquad \text{for all } i \in \{1,\ldots,p\}, t' \in \mathcal{T}_\mathcal{I},$$

and this implies

$$\#\widehat{N}_t = \sum_{i=1}^{p} \#\widetilde{N}_{i,t} \leq \sum_{i=1}^{p} k_{i,t} \qquad \text{for all } t \in \mathcal{T}_\mathcal{I}.$$

Since $\widehat{X}_t$ has only $\#\widehat{N}_t$ columns, its rank cannot be higher, therefore also the rank of the new cluster basis Q_t has to be bounded by

$$\#\widetilde{K}_t \leq \#\widehat{N}_t \leq \sum_{i=1}^{p} k_{i,t} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}}. \tag{3.11}$$

Let us now consider the last remaining quantity $\#\widehat{K}_t$. If t is a leaf, we have $\widehat{K}_t = \hat{t}$ and get $\#\widehat{K}_t \leq C_{\mathrm{lf}} k_{i,t}$ for any $i \in \{1, \ldots, p\}$, which trivially implies

$$\#\widehat{K}_t \leq C_{\mathrm{lf}} \sum_{i=1}^{p} k_{i,t} \qquad \text{for all } t \in \mathcal{L}_{\mathcal{I}}.$$

Otherwise, we have

$$\#\widehat{K}_t = \sum_{t' \in \operatorname{sons}(t)} \#\widetilde{K}_{t'} \leq \sum_{i=1}^{p} \sum_{t' \in \operatorname{sons}(t)} k_{i,t'} \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}} \setminus \mathcal{L}_{\mathcal{I}}.$$

Using these preliminary estimates, we can now give a bound for the algorithmic complexity of Algorithm 3:

LEMMA 3.2 (Complexity of unification). *There is a constant $C_{\mathrm{uni}} \in \mathbb{R}_{>0}$ depending only on C_{qr}, C_{svd}, C_{lf} and C_{sn} such that Algorithm 3 requires not more than*

$$C_{\mathrm{uni}} p^2 \sum_{i=1}^{p} \sum_{r \in \mathcal{T}_{\mathcal{I}}^t} k_{i,r}^3 \quad \textit{arithmetic operations.}$$

Proof. Let $t \in \mathcal{T}_{\mathcal{I}}$. Algorithm 3 starts by preparing the matrix $\widehat{V}_{i,t}$. If t is a leaf, the matrix $V_{i,t}$ is copied and no arithmetic operations are performed. If t is not a leaf, the matrices $R_{i,t'}$ and $E_{i,t'}$ are multiplied for all $i \in \{1, \ldots, p\}$ and $t' \in \operatorname{sons}(t)$, and this requires not more than

$$\sum_{i=1}^{p} \sum_{t' \in \operatorname{sons}(t)} 2(\#\widetilde{K}_{t'})(\#K_{i,t'})(\#K_{i,t}) \leq 2 \sum_{i=1}^{p} \sum_{j=1}^{p} \sum_{t' \in \operatorname{sons}(t)} k_{j,t'} k_{i,t'} k_{i,t} \quad \text{operations}$$

due to (3.11). We employ the elementary inequality

$$xyz \leq \frac{1}{3}(x^3 + y^3 + z^3) \qquad \text{for all } x, y, z \in \mathbb{R}_{\geq 0}$$

to bound this term by

$$\frac{2}{3} \sum_{i=1}^{p} \sum_{j=1}^{p} \sum_{t' \in \operatorname{sons}(t)} k_{j,t'}^3 + k_{i,t'}^3 + k_{i,t}^3 \leq \frac{2}{3} p \sum_{i=1}^{p} \sum_{t' \in \operatorname{sons}(t)} 2k_{i,t'}^3 + k_{i,t}^3.$$

Since most of our estimates will be of a similar form with different constants, we introduce

$$\alpha_t := \sum_{i=1}^{p} k_{i,t}^3, \qquad \beta_t := \sum_{i=1}^{p} \sum_{t' \in \operatorname{sons}(t)} k_{i,t'}^3$$

and get

$$\sum_{i=1}^{p} \sum_{t' \in \operatorname{sons}(t)} 2(\#\widetilde{K}_{t'})(\#K_{i,t'})(\#K_{i,t}) \leq \left(\frac{2}{3} C_{\mathrm{sn}} p\right) \alpha_t + \left(\frac{4}{3} p\right) \beta_t.$$

We can use similar techniques to prove that the number of operations for the construction of $\widehat{X}_t$ by multiplying $\widehat{V}_{i,t}$ and $\widetilde{Z}_{i,t}$ is bounded by

$$\sum_{i=1}^{p} 2(\#\widehat{K}_t)(\#K_{i,t})(\#\widetilde{N}_{i,t}) \leq \left(2C_{\mathrm{lf}} p + \frac{4}{3} C_{\mathrm{sn}} p\right) \alpha_t + \left(\frac{2}{3} p\right) \beta_t,$$

that the computation of the singular value decomposition of $\widehat{X}_t$ takes not more than

$$C_{\mathrm{svd}}(\#\widehat{K}_t)(\#\widehat{N}_t)^2 \leq \left(C_{\mathrm{svd}} C_{\mathrm{lf}} p^2 + \frac{2}{3} C_{\mathrm{svd}} C_{\mathrm{sn}} p^2\right) \alpha_t + \left(\frac{1}{3} C_{\mathrm{svd}} p^2\right) \beta_t$$

operations, that the transformation matrices $R_{i,t}$ can be constructed in not more than

$$2 \sum_{i=1}^{p} (\#\widetilde{K}_t)(\#\widehat{K}_t)(\#K_{i,t}) \leq \left(2C_{\mathrm{lf}}^2 p^2 + \frac{2}{3} C_{\mathrm{sn}} p^2\right) \alpha_t + \left(\frac{4}{3} p^2\right) \beta_t$$

operations, that we can compute the "uncondensed" weight matrix $\widetilde{Y}_t$ in not more than

$$2(\#\widetilde{K}_t)(\#\widehat{K}_t)(\#\widehat{N}_t) \leq \left(2C_{\mathrm{lf}}^2 p^2 + \frac{2}{3} C_{\mathrm{sn}} p^2\right) \alpha_t + \left(\frac{4}{3} p^2\right) \beta_t$$

operations and that the orthogonal factorization used to find the final weight matrix $\widetilde{Z}_t$ takes not more than

$$C_{\mathrm{qr}}(\#\widehat{N}_t)(\#\widetilde{K}_t)^2 \leq \left(C_{\mathrm{qr}} C_{\mathrm{lf}}^2 p^2 + \frac{1}{3} C_{\mathrm{qr}} C_{\mathrm{sn}} p^2\right) \alpha_t + \left(\frac{2}{3} C_{\mathrm{qr}} p^2\right) \beta_t$$

arithmetic operations. Adding up these estimates yields a bound of

$$C_1 p^2 \alpha_t + C_2 p^2 \beta_t = C_1 p^2 \sum_{i=1}^{p} k_{i,t}^3 + C_2 p^2 \sum_{i=1}^{p} \sum_{t' \in \operatorname{sons}(t)} k_{i,t'}^3$$

for constants $C_1, C_2 \in \mathbb{R}_{\geq 0}$ depending only on C_{qr}, C_{svd}, C_{lf} and C_{sn}.

Since Algorithm 3 uses recursive calls to compute the matrices also for all descendants r of t, a bound for the total number of operations can be derived by summing this result over all descendants. Due to the fact that each of these descendants cannot have more than one father, we get the bound

$$\begin{aligned}\sum_{r \in \mathcal{T}_{\mathcal{I}}^t} C_1 p^2 \alpha_t + C_2 p^2 \beta_t &= C_1 p^2 \sum_{i=1}^{p} \sum_{r \in \mathcal{T}_{\mathcal{I}}^t} k_{i,r}^3 + C_2 p^2 \sum_{i=1}^{p} \sum_{r \in \mathcal{T}_{\mathcal{I}}^t} \sum_{r' \in \operatorname{sons}(r)} k_{i,r'}^3 \\ &\leq C_1 p^2 \sum_{i=1}^{p} \sum_{r \in \mathcal{T}_{\mathcal{I}}^t} k_{i,r}^3 + C_2 p^2 \sum_{i=1}^{p} \sum_{r' \in \mathcal{T}_{\mathcal{I}}^t} k_{i,r'}^3\end{aligned}$$

$$\leq (C_1 + C_2)p^2 \sum_{i=1}^{p} \sum_{r \in \mathcal{T}_{\mathcal{I}}^t} k_{i,r}^3,$$

and using $C_{\mathrm{uni}} := C_1 + C_2$, this is the estimate we need. □

We can again consider special cases: if $k_{i,t} \leq k$ holds for all $i \in \{1, \ldots, p\}$ and all $t \in \mathcal{T}_{\mathcal{I}}$, Algorithm 3 requires $\mathcal{O}((\#\mathcal{T}_{\mathcal{I}})k^3p^3)$ operations, and for $\#\mathcal{T}_{\mathcal{I}} \lesssim n/k$, we get $\mathcal{O}(nk^2p^3)$. Lemma 3.2 provides us with a worst-case bound: our estimates allow the case $\#\widetilde{K}_t = \#\widehat{N}_t$, corresponding to completely unrelated submatrices, while in practice we expect that the submatrices correspond to subblocks of a matrix that can be approximated globally by an $\mathcal{H}^2$-matrix, therefore the resulting rank $\#\widetilde{K}_t$ can be expected to be similar to $\#K_{i,t}$.

3.7. Hierarchical compression. Based on the unification technique we can now introduce the hierarchical compression algorithm.

A simple approach is to use the block cluster tree: for each admissible leaf, we construct a low-rank approximation of the corresponding submatrix using one of the established $\mathcal{H}$-matrix techniques (e.g., matrix arithmetics [20, 17] or cross approximation [3, 8]). Then we work towards the root of the block cluster tree and construct unified row and column cluster bases for each submatrix.

Each unification adds an error bounded by ϵ to the final result, leading to a total error that can be bounded by $\epsilon \operatorname{depth}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}})$, and therefore is under control.

The analysis of the complexity of the hierarchical compression scheme is a little more complicated: the number of operations required for constructing the leaf blocks $\mathcal{L}_{\mathcal{I}\times\mathcal{J}}$ depends on the original approximation scheme, and we can apply standard results [17] of the $\mathcal{H}$-matrix theory. The number of operations required for conversion into the $\mathcal{H}^2$-matrix format can be bounded using Lemma 3.2: Algorithm 3 has to be applied for all $b = (t, s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}} \setminus \mathcal{L}_{\mathcal{I}\times\mathcal{J}}$, and assuming that all ranks can be bounded by a constant $k \in \mathbb{N}$, Lemma 3.2 yields a bound of

$$\sum_{\substack{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}} \\ b\notin\mathcal{L}_{\mathcal{I}\times\mathcal{J}}}} C_{\mathrm{uni}}p^3 \left(\sum_{r\in\mathcal{T}_{\mathcal{I}}^t} k^3 + \sum_{r\in\mathcal{T}_{\mathcal{J}}^s} k^3 \right) \quad \text{arithmetic operations.}$$

Assuming again that $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ is sparse, we get

$$\begin{aligned}
\sum_{\substack{b=(t,s)\in\mathcal{T}_{\mathcal{I}\times\mathcal{J}} \\ b\notin\mathcal{L}_{\mathcal{I}\times\mathcal{J}}}} C_{\mathrm{uni}}p^3 \left(\sum_{r\in\mathcal{T}_{\mathcal{I}}^t} k^3 + \sum_{r\in\mathcal{T}_{\mathcal{J}}^s} k^3 \right) &\leq C_{\mathrm{sp}}C_{\mathrm{uni}}p^3 \left(\sum_{t\in\mathcal{T}_{\mathcal{I}}} \sum_{r\in\mathcal{T}_{\mathcal{I}}^t} k^3 + \sum_{s\in\mathcal{T}_{\mathcal{J}}} \sum_{r\in\mathcal{T}_{\mathcal{J}}^s} k^3 \right) \\
&= C_{\mathrm{sp}}C_{\mathrm{uni}}p^3 \left(\sum_{r\in\mathcal{T}_{\mathcal{I}}} k^3 \#\operatorname{pred}(r) + \sum_{r\in\mathcal{T}_{\mathcal{J}}} k^3 \#\operatorname{pred}(r) \right) \\
&\leq C_{\mathrm{sp}}C_{\mathrm{uni}}p^3k^3 \left(\operatorname{depth}(\mathcal{T}_{\mathcal{I}})\#\mathcal{T}_{\mathcal{I}} + \operatorname{depth}(\mathcal{T}_{\mathcal{J}})\#\mathcal{T}_{\mathcal{J}} \right).
\end{aligned}$$

Under the standard assumptions $\operatorname{depth}(\mathcal{T}_{\mathcal{I}}), \operatorname{depth}(\mathcal{T}_{\mathcal{J}}) \lesssim \log n$ and $\#\mathcal{T}_{\mathcal{I}}, \#\mathcal{T}_{\mathcal{J}} \lesssim n/k$, we conclude that the hierarchical compression requires $\mathcal{O}(nk^2 \log n)$ operations. This is the same order of complexity as for most $\mathcal{H}$-matrix algorithms, i.e., the conversion to the more efficient $\mathcal{H}^2$-matrix format can be performed without changing the complexity of the underlying $\mathcal{H}$-matrix algorithm.

	SLP matrix			DLP matrix			L^2 error		
n	Build	Mem	M/n	Build	Mem	M/n	ϵ_1	ϵ_2	ϵ_3
2048	6.7	8.7	4.4	12.5	7.4	3.7	1.2_{-1}	2.3_{-2}	1.8_{-1}
8192	36.9	48.5	6.1	64.3	38.5	4.8	6.2_{-2}	1.1_{-2}	8.9_{-2}
32768	205.7	232.4	7.3	310.6	177.6	5.6	3.1_{-2}	5.6_{-3}	4.4_{-2}
131072	1069.7	1257.0	9.8	1706.2	1034.6	8.1	1.5_{-2}	2.8_{-3}	2.2_{-2}
524288	6051.3	5529.0	10.8	8152.7	4552.1	8.9	7.7_{-3}	1.4_{-3}	1.1_{-2}

TABLE 4.1
Boundary integral operators on the unit sphere

4. Numerical experiments. We apply the hierarchical compression scheme to a boundary integral problem: consider a Lipschitz domain $\Omega \subseteq \mathbb{R}^3$ and a harmonic function u in this domain. The Dirichlet values $u|_\Gamma$ on the boundary $\Gamma := \partial\Omega$ of Ω are connected to the Neumann values $\partial_n u|_\Gamma$ by Green's formula

$$\int_\Gamma g(x,y)\partial_n u(y)\,dy = \frac{1}{2}u(x) + \int_\Gamma \partial_{n(y)} g(x,y)u(y)\,dy \qquad \text{for all } x \in \Gamma, \tag{4.1}$$

which allows us to compute the Neumann values corresponding to given Dirichlet values, where the kernel function is given by

$$g(x,y) := \frac{1}{4\pi}\frac{1}{\|x-y\|_2} \qquad \text{for all } x, y \in \mathbb{R}^3, x \neq y.$$

The integral operator on the left-hand side of (4.1) is called the single layer potential (SLP) operator, the integral operator on the right-hand side is called the double layer potential (DLP) operator. We discretize (4.1) by a Galerkin scheme with piecewise constant basis functions for the Neumann values $\partial_n u|_\Gamma$ and continuous piecewise linear basis functions for the Dirichlet values $u|_\Gamma$, thus approximating the single and double layer potential operators by matrices V and K.

These matrices are dense and can only be handled efficiently if a compression scheme is applied. We use the unification Algorithm 3 in combination with an initial low-rank approximation provided by the HCA method [8] and a grey-box quadrature rule [26]. Strang's lemma (e.g., [11, Theorem 4.1.1]) implies that error estimates of the form $\|V - \widetilde{V}\|_2 \lesssim h^4$ and $\|K - \widetilde{K}\|_2 \lesssim h^4$ would ensure that the optimal order of convergence of the overall scheme is preserved. We achieve this goal by using the advanced error control technique presented in [5].

Table 4.1 contains the results for a simple situation: we approximate the unit sphere by n plane triangles and apply our scheme to the harmonic functions

$$u_1(x) = x_1^2 - x_3^2, \quad u_2(x) = \frac{1}{\|x-x^*\|_2}, \quad u_3(x) = \frac{1}{\|x-x^{**}\|_2} \qquad \text{for all } x \in \mathbb{R}^3$$

with $x^* = (1.2, 1.2, 1.2)$ and $x^{**} = (1, 1/4, 1)$ to test the approximation properties of the approach. The columns "Build" contain the time for the construction of the matrices (including quadrature of the singular nearfield integrals) in seconds, measured on one processor of a SunFire X4600 computer, the columns "Mem" give the storage requirements for near- and farfield in MBytes, the columns "M/n" give the

	$\mathcal{H}^2$-matrix				$\mathcal{H}$-matrix			
			Spectral error				Spectral error	
n	Build	M/n	abs.	rel.	Build	M/n	abs.	rel.
25744	166.7	7.7	1.2_{-8}	5.9_{-5}	138.1	7.4	1.8_{-8}	8.9_{-5}
102976	807.4	7.9	6.1_{-10}	1.2_{-5}	636.3	12.2	9.3_{-10}	1.9_{-5}
411904	4814.7	7.7	2.5_{-11}	2.1_{-6}	3831.9	18.8	2.8_{-11}	2.2_{-6}
1647616	25751.0	8.3	1.4_{-12}	4.5_{-7}	21646.1	27.6	2.0_{-12}	6.4_{-7}
25088	170.5	7.9	1.2_{-8}	1.6_{-5}	141.4	9.0	1.7_{-8}	2.2_{-5}
100352	886.3	9.6	5.2_{-10}	2.7_{-6}	717.4	13.4	6.1_{-10}	3.2_{-6}
401408	4479.1	10.1	2.7_{-11}	5.6_{-7}	3628.8	20.1	3.5_{-11}	7.2_{-7}
1605632	28257.2	10.8	1.4_{-12}	1.1_{-7}	25784.1	29.5	2.4_{-12}	2.0_{-7}
28952	266.1	15.1	7.2_{-8}	1.6_{-5}	237.3	10.5	8.1_{-8}	1.9_{-5}
115808	1133.2	12.1	3.8_{-9}	3.4_{-6}	952.8	16.8	5.8_{-9}	5.2_{-6}
463232	5949.8	12.7	2.3_{-10}	8.1_{-7}	4959.4	26.0	2.8_{-10}	9.8_{-7}
1852928	34921.5	9.5	1.4_{-11}	2.0_{-7}	33409.8	40.3	2.1_{-11}	3.0_{-7}

TABLE 4.2
Comparison of $\mathcal{H}$- and $\mathcal{H}^2$-matrix compression

storage requirements per degree of freedom in KBytes, while ϵ_1, ϵ_2 and ϵ_2 are the L^2-norm errors of the approximated Neumann values. The storage requirements for the SLP matrix $\widetilde{V}$ could be reduced further by taking advantage of its symmetry without changing the accuracy, but this is not yet implemented. We can see that the errors converge like $1/n$, which is the optimal rate for a piecewise constant approximation.

In a second experiment, we compare the $\mathcal{H}^2$-matrix approximation provided by the hierarchical compression algorithm with an approximation in the $\mathcal{H}$-matrix format: the adaptive coarsening algorithm [15] not only uses near-optimal low-rank approximations in each admissible block, it also optimizes the block cluster tree in order to reduce the storage requirements. The resulting $\mathcal{H}$-matrix is very close to the best possible approximation in this representation.

We compare both techniques using three different geometries: the first two are examples from the NetGen package by Joachim Schöberl, namely an approximation of a crank shaft with 25744 plane triangles and an approximation of a pierced sphere with 25088 plane triangles, the third one is an approximation of a three-dimensional foam with 28952 plane triangles courtesy of Günther Of and Heiko Andrä. Each of the geometries is refined three times by splitting each triangle into four congruent subtriangles, thus providing us with a range of problem dimensions.

We approximate the single layer potential matrix V on each of the resulting twelve surface meshes and compare the time for the construction and the storage requirements. The results are given in Table 4.2, and we can see that the time requirements for the construction of $\mathcal{H}$- and $\mathcal{H}^2$-matrices are comparable and that the compression works very well for large problem dimensions: for 100000 degrees of freedom, both representations require similar amount of storage, for 400000 the $\mathcal{H}^2$-matrix requires less than half the amount needed by the $\mathcal{H}$-matrix, and for 1.6 million the ratio is less than one third.

The experiments demonstrate that with the hierarchical compression algorithm,

the $\mathcal{H}^2$-matrix method is significantly better than the best known $\mathcal{H}$-matrix approach: it requires approximately the same amount of time (which is no surprise, since it is based on the same initial low-rank approximation), and its improved storage complexity is clearly visible for large problem dimensions.

REFERENCES

[1] M. Bebendorf, *Approximation of boundary element matrices*, Numer. Math., 86 (2000), pp. 565–589.

[2] M. Bebendorf and W. Hackbusch, *Existence of $\mathcal{H}$-matrix approximants to the inverse FE-matrix of elliptic operators with L^∞-coefficients*, Numerische Mathematik, 95 (2003), pp. 1–28.

[3] M. Bebendorf and S. Rjasanow, *Adaptive Low-Rank Approximation of Collocation Matrices*, Computing, 70 (2003), pp. 1–24.

[4] S. Börm, *Approximation of integral operators by $\mathcal{H}^2$-matrices with adaptive bases*, Computing, 74 (2005), pp. 249–271.

[5] ———, *Adaptive variable-rank approximation of dense matrices*, SIAM Journal on Scientific Computing, 30 (2007), pp. 148–168.

[6] ———, *Approximation of solution operators of elliptic partial differential equations by $\mathcal{H}$- and $\mathcal{H}^2$-matrices*, Tech. Report 85, Max Planck Institute for Mathematics in the Sciences, Leipzig, 2007.

[7] ———, *Data-sparse approximation of non-local operators by $\mathcal{H}^2$-matrices*, Linear Algebra and its Applications, 422 (2007), pp. 380–403.

[8] S. Börm and L. Grasedyck, *Hybrid cross approximation of integral operators*, Numerische Mathematik, 101 (2005), pp. 221–249.

[9] S. Börm, L. Grasedyck, and W. Hackbusch, *Hierarchical Matrices*. Lecture Note 21 of the Max Planck Institute for Mathematics in the Sciences, 2003.

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

[11] P. G. Ciarlet, *The finite element method for elliptic problems*, SIAM, 2002.

[12] I. Gavrilyuk, W. Hackbusch, and B. N. Khoromskij, *$\mathcal{H}$-matrix approximation for the operator exponential with applications*, Numerische Mathematik, 92 (2002), pp. 83–111.

[13] ———, *Data-sparse approximation to operator-valued functions of elliptic operator*, Mathematics of Computation, 73 (2004), pp. 1107–1138.

[14] L. Grasedyck, *Theorie und Anwendungen Hierarchischer Matrizen*, doctoral thesis, Universität Kiel, 2001.

[15] ———, *Adaptive recompression of $\mathcal{H}$-matrices for BEM*, Computing, 74 (2004), pp. 205–223.

[16] Lars Grasedyck, *Existence of a low-rank or $\mathcal{H}$-matrix approximant to the solution of a Sylvester equation*, Num. Lin. Alg. Appl., 11 (2004), pp. 371–389.

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

[18] L. Grasedyck, W. Hackbusch, and B. N. Khoromskij, *Solution of large scale algebraic matrix Riccati equations by use of hierarchical matrices*, Computing, 70 (2003), pp. 121–165.

[19] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, Journal of Computational Physics, 73 (1987), pp. 325–348.

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

[21] 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.

[22] W. Hackbusch, B. N. Khoromskij, and S. A. Sauter, *On $\mathcal{H}^2$-matrices*, in Lectures on Applied Mathematics, H. Bungartz, R. Hoppe, and C. Zenger, eds., Springer-Verlag, Berlin, 2000, pp. 9–29.

[23] W. Hackbusch and Z. P. Nowak, *On the fast matrix multiplication in the boundary element method by panel clustering*, Numerische Mathematik, 54 (1989), pp. 463–491.

[24] V. Rokhlin, *Rapid solution of integral equations of classical potential theory*, Journal of Computational Physics, 60 (1985), pp. 187–207.

[25] S. A. Sauter, *Variable order panel clustering*, Computing, 64 (2000), pp. 223–261.

[26] S. A. Sauter and C. Schwab, *Randelementmethoden*, Teubner, 2004.