Max-Planck-Institut
für Mathematik
in den Naturwissenschaften
Leipzig

Data-sparse approximation by adaptive $\mathbf{H^2}$-Matrices

by

Wolfgang Hackbusch and Steffen Börm

Preprint no.: 86 2001

Data-sparse Approximation by Adaptive $\mathcal{H}^2$-Matrices

Wolfgang Hackbusch, Steffen Börm
Max-Planck-Institut *Mathematik in den Naturwissenschaften*
Inselstr. 22-26, D-04103 Leipzig, Germany
email: {wh,sbo}@mis.mpg.de

November 15, 2001

A class of matrices ($\mathcal{H}^2$-matrices) has recently been introduced for storing discretisations of elliptic problems and integral operators from the BEM. These matrices have the following properties: (i) They are sparse in the sense that only few data are needed for their representation. (ii) The matrix-vector multiplication is of linear complexity. (iii) In general, sums and products of these matrices are no longer in the same set, but after truncation to the $\mathcal{H}^2$-matrix format these operations are again of quasi-linear complexity.

We introduce the basic ideas of $\mathcal{H}$- and $\mathcal{H}^2$-matrices and present an algorithm that adaptively computes approximations of general matrices in the latter format.

AMS Subject Classifications: 65F05, 65F30, 65F50, 65N38, 68P05, 45B05, 35C20
Key words: Hierarchical matrices, nested bases, full matrices, fast matrix-vector multiplication, BEM, FEM.

1 Introduction

For linear systems with a sparse $n \times n$-matrix A, several optimal iteration methods are known, where optimality is characterised by estimating the arithmetic operations by $\mathcal{O}(n)$. Usually, it is a well-established rule to avoid any consideration of A^{-1} for numerical purposes because of the fill-in (usually, A^{-1} is a full matrix). Under this restriction, one tries to express all algorithmical steps by means of the non-zero entries of A or by the action of the matrix A to a vector. ILU-decompositions or Krylov methods are typical results. A particular problem arises for the iterative solution of $Sx = y$, where S is a Schur complement. Since the exact calculation of S is "forbidden" because it involves the inversion of a submatrix, only the action of S can be considered. There are cases where this is not sufficient for constructing a fast iterative method.

The techniques introduced in [4] show that, nevertheless, one can handle full matrices like A^{-1} with (almost) linear cost. In particular, in [6] a class of matrices (called $\mathcal{H}^2$-*matrices*) was introduced for discrete elliptic problems and discrete integral operators from the boundary element method (BEM, cf. [3]). In the latter case, full matrices arise directly. It is shown in [6] that $\mathcal{H}^2$-matrices have the following properties:

1. They are data-sparse in the sense that only $\mathcal{O}(n)$ data are needed for their representation.
2. The matrix-vector multiplication is of linear complexity.
3. Sums of these matrices can be computed with linear complexity.

In the case of sparse matrices A resulting from finite element discretisations of elliptic boundary value problems, the inverse A^{-1} approximated by the $\mathcal{H}^2$-format is the interesting object. Depending on the accuracy of the approximation $B \approx A^{-1}$, we have (a) a new fast iteration $x^{i+1} := x^i - B(Ax^i - y)$, if B is a rough approximation to A^{-1}, or (b) By can be considered as *the* discrete solution, if the error $B - A^{-1}$ is of the size of the discretisation error for y.

The notation $\mathcal{H}^2$ refers to two different hierarchical structures. The (first) hierarchy of the underlying *cluster tree* leads to the definition of the $\mathcal{H}$-matrices introduced in detail in [4] and [5]. A second hierarchy concerns the nested bases used in the definition of $\mathcal{H}^2$-matrices.

In Section 3, we review the definition of $\mathcal{H}^2$-matrices. The following Section 4 considers the efficient implementation of some basic operations on this type of matrices.

The computation of an optimal $\mathcal{H}$-approximation of a given general matrix can be done in principle by using the singular value decomposition for all matrix blocks and requires $\mathcal{O}(n^3)$ operations. This trivial but costly algorithm no longer works if an $\mathcal{H}^2$-approximation is desired. In Section 5, we present an algorithm that computes such an approximation with only $\mathcal{O}(n^2)$ operations. If the original matrix is given in $\mathcal{H}$-format, the algorithm can be modified to use only an almost linear amount of operations.

2 $\mathcal{H}$-Matrices

In Subsections 2.1 and 2.2, we define the cluster tree and block cluster tree which, together with an admissibility condition for matrix blocks described in Subsection 2.3, form the basis of the $\mathcal{H}$-matrix format introduced in Subsection 2.4. An introductory example is presented in Subsection 2.5. These preparations will lead to the nested bases $\mathcal{H}$-matrices ($\mathcal{H}^2$-matrices) in Section 3.

2.1 The Cluster Tree

We use a tree structure with the following notations: A tree $\mathcal{T}(I)$ is described by a set of *nodes* $T(I)$ and a mapping $S : T(I) \to \mathcal{P}(T(I))$ satisfying

$$S(\tau) := \{\sigma \in T(I) : \sigma \text{ is a son of } \tau\} \qquad \text{for } \tau \in T(I),$$

i.e., $S(\tau)$ is the *set of sons* of a given node τ.

If for $\tau \in T(I)$ the equation $S(\tau) = \emptyset$ holds, τ is called a *leaf*. The set of leaves is denoted by $\mathcal{L}(\mathcal{T}(I))$.

Let I be the index set of the finitely many degrees of freedom. The elements of I may be $\{1, \ldots, n\}$ or the nodal points of the finite element discretisation, etc. The precise definition of the *cluster tree* $\mathcal{T}(I)$ is given in

Definition 2.1 *Let I be an index set. A tree $\mathcal{T}(I)$ is called a* cluster tree *(based on I) if the following conditions hold:*

1. *I is the root of $\mathcal{T}(I)$.*
2. *$\#S(\tau) \neq 1$ for all $\tau \in T(I)$.*
3. *For all $\tau \in T(I)$, we have*

$$S(\tau) \neq \emptyset \Rightarrow \tau = \dot{\bigcup}_{\tau' \in S(\tau)} \tau', \tag{2.1}$$

   i.e., if τ is not a leaf, $S(\tau)$ is a partition of τ.

4. *There is a constant $C_L \in \mathbb{N}$ independent from I satisfying*

$$S(\tau) = \emptyset \Rightarrow n_\tau \leq C_L \tag{2.2}$$

   for all $\tau \in T(I)$ with $n_\tau := \#\tau$, i.e., the size of each leaf is bounded.

For technical reasons, condition 2 is sometimes omitted, but then two different vertices of $T(I)$ may correspond to identical subsets of I (cf. (2.1)).

We define

$$\begin{gathered} \operatorname{level} : T(I) \to \mathbb{N}_0 \quad \text{by} \\ \operatorname{level}(I) := 0 \quad \text{and} \quad \operatorname{level}(\tau') := \operatorname{level}(\tau) + 1 \quad \text{for } \tau' \in S(\tau). \end{gathered}$$

The maximal level of the cluster tree $\mathcal{T}(I)$ is $\ell_{\max} := \max \operatorname{level}(T(I))$, i.e., the depth of $\mathcal{T}(I)$. $T(I)$ can be partitioned into

$$T(I) = \bigcup_{\ell=0}^{\ell_{\max}} T_\ell(I),$$

where $T_\ell(I) := \{\tau \in T(I) \ : \ \operatorname{level}(\tau) = \ell\}$ for $\ell \in \{0, \ldots, \ell_{\max}\}$.

2.2 The Block Cluster Tree

Since we are dealing with matrices $A = (a_{ij})_{i,j \in I}$, we have to consider the index set $I \times I$. A hierarchical partitioning of this set is given by the *block cluster tree* $\mathcal{T}(I \times I)$ (with nodes $T(I \times I)$) that is derived from $\mathcal{T}(I)$ by the following inductive construction:

Construction 2.2 *The root of $\mathcal{T}(I \times I)$ is $I \times I$. Given a vertex $b = \tau \times \sigma \in T(I \times I)$ with $\tau, \sigma \in T(I)$, the set of sons of b is defined to be*

$$S(b) := \{\tau' \times \sigma' \ : \ \tau' \in S(\tau), \sigma' \in S(\sigma)\}. \tag{2.3}$$

Note that b is a leaf if either τ or σ is a leaf.

Remark 2.3 *A binary cluster tree $\mathcal{T}(I)$ leads to a quad-tree $\mathcal{T}(I \times I)$.*

The mapping level$(\cdot)$ defined on $T(I)$ is extended to $T(I \times I)$ as explained in

Remark 2.4 *For all $b = \tau \times \sigma \in T(I \times I)$, we define* $\mathrm{level}(b) = \mathrm{level}(\tau)$. *Due to* (2.3), *we also have* $\mathrm{level}(b) = \mathrm{level}(\sigma)$. *We can partition $T(I \times I)$ into*

$$T(I \times I) = \bigcup_{\ell=0}^{\ell_{\max}} T_\ell(I \times I),$$

where $T_\ell(I \times I) = \{b \in T(I \times I) \ : \ \mathrm{level}(b) = \ell\}$ for all $\ell \in \{0, \ldots, \ell_{\max}\}$.

2.3 Admissibility

Each index $i \in I$ is associated with a subset $X_i \subset \mathbb{R}^d$. This may be a grid point (i.e., $X_i = \{x_i\}$) or the support of the ith finite element basis function:

$$X_i := \mathrm{supp}(b_i) \qquad (b_i : i\,\mathrm{th\ finite\ element\ basis\ function}). \tag{2.4}$$

This notation is extended to clusters $\tau \in T(I)$ by

$$X_\tau := \bigcup\nolimits_{i \in \tau} X_i. \tag{2.5}$$

We introduce the *diameter* and the *distance* of $\tau, \sigma \in T(I)$ by setting

$$\mathrm{diam}(\tau) := \max_{x,y \in X_\tau} \|x - y\| \quad \text{and} \quad \mathrm{dist}(\tau, \sigma) := \min_{x \in X_\tau, y \in X_\sigma} \|x - y\|.$$

A block $b = \tau \times \sigma \in T(I \times I)$ is called *admissible*, if b is a leaf or if the following *admissibility condition* [1] holds for a fixed parameter $\eta < 1$:

$$\max\{\mathrm{diam}(\tau), \mathrm{diam}(\sigma)\} \leq 2\eta \ \mathrm{dist}(\tau, \sigma). \tag{2.6}$$

[1] For general $\mathcal{H}$-matrices the maximum on the left-hand side in (2.6) can be replaced by the minimum. However, for some proofs in the context of $\mathcal{H}^2$-matrices we need the maximum.

2.4 Admissible Matrix Partitionings and $\mathcal{H}$-Matrices

We introduce the set-valued functions

$$\begin{aligned} \operatorname{cols} : T(I) \times \mathcal{P}(T(I) \times T(I)) &\to \mathcal{P}(T(I)), \\ (\tau, P) &\mapsto \{\sigma \in T(I) \ : \ \tau \times \sigma \in P\}, \\ \operatorname{rows} : T(I) \times \mathcal{P}(T(I) \times T(I)) &\to \mathcal{P}(T(I)), \\ (\sigma, P) &\mapsto \{\tau \in T(I) \ : \ \tau \times \sigma \in P\}, \end{aligned}$$

that associate a given cluster with all blocks in which it occurs.

A set $P_2 \in \mathcal{P}(T(I) \times T(I))$ satisfying

$$I \times I = \dot{\bigcup} \ \{b \ : \ b \in P_2\} \quad \text{and} \quad P_2 \subseteq T(I \times I) \tag{2.7}$$

is called a *hierarchical partitioning* of $I \times I$. The subscript 2 in P_2 should indicate that P_2 partitions the *two*fold product index set $I \times I$. Note that the second condition implies $\operatorname{level}(\tau) = \operatorname{level}(\sigma)$ for all $\tau, \sigma \in T(I)$ with $\tau \times \sigma \in P_2$.

For any P_2 there exists exactly one subtree T_2 of $T(I \times I)$ with root $I \times I$ satisfying

$$S(b) = \emptyset \iff b \in P_2 \quad \text{for all } b \in T_2. \tag{2.8}$$

A partitioning P_2 is called *admissible* if all blocks $b \in P_2$ are admissible.

Definition 2.5 ($\mathcal{H}$-matrix) *Let I be an index set, let P_2 be a block partitioning of $I \times I$ and let $k \in \mathbb{N}^{P_2}$. A matrix $A \in \mathbb{K}^{I \times I}$ is an* $\mathcal{H}$-matrix with respect to P_2 and the rank distribution k, *if for each $b \in P_2$ the condition* $\operatorname{rank}(A|_b) \leq k_b$ *holds.*

Here, $A|_b = (a_{ij})_{(i,j) \in b}$ denotes the block matrix with respect to $b \in P_2$.

The set of all these matrices is denoted by $\mathcal{M}_{\mathcal{H}}(I \times I, P_2, k)$.

2.5 Example

To fix the ideas, we give a 1D example of what an admissible partitioning may look like. We consider piecewise constant finite elements on the unit interval $[0, 1]$ having the supports $X_i = [(i-1)h, ih]$ for $i = 1, \ldots, n$ with the step size $h = 1/n$. Assume $n = 2^p$. Then an obvious recursive construction of the cluster tree is as follows: $I = \{1, \ldots, n\}$ is the root, and whenever

$$\tau = \{i_\tau, i_\tau + 1, \ldots, i_\tau + n_\tau - 1\} \tag{2.9}$$

is not a leaf (i.e., $n_\tau > C_L$, cf. (4)), the sons of τ are the two clusters $\tau' = \{i_\tau, i_\tau + 1, \ldots, i_\tau + n_\tau/2 - 1\}$ and $\tau'' = \{i_\tau + n_\tau/2, \ldots, i_\tau + n_\tau - 1\}$ of half the size. An equivalent way would be to say that the support X_τ is divided into equal parts $X_{\tau'}$ and $X_{\tau''}$. We have $\tau \in T_l(I)$ if and only if $n_\tau = 2^{p-l}$, where $p = \ell_{\max}$. Furthermore, $\operatorname{diam}(\tau) = 2^{p-l}h = 2^{-l}$ and $\operatorname{dist}(\tau, \sigma) = \max\{0, |i_\tau - i_\sigma|h - 2^{-l}\}$ for $\tau, \sigma \in T_l(I)$, where the notation i_τ refers to (2.9).

The partitioning P_2 defined by the admissibility condition (2.6) for $\eta = 1/2$ takes the form outlined in Figure 1.

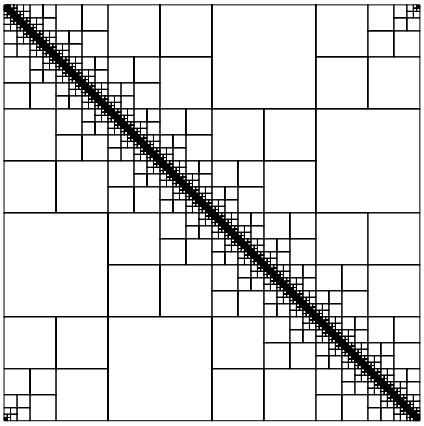

Figure 1: Admissible partitioning

3 $\mathcal{H}^2$-Matrices

The $\mathcal{H}^2$-matrices, which we are going to explain in this section, were first introduced in [6]. They are a subset of the general $\mathcal{H}$-matrices $\mathcal{M}_{\mathcal{H}}(I \times I, P_2, k)$. For $A \in \mathcal{M}_{\mathcal{H}}(I \times I, P_2, k)$, we allow $A|_b$ to be *any* block-matrix of rank $\leq k_b$.

Now we will restrict $A|_b$ to a certain subspace $\mathcal{V}_b$ of matrices with rank $\leq k_b$ in order to reduce the amount of work necessary for operations on this new type of matrix. This restriction can be seen as a three-step procedure: The first step is to consider arbitrary, but fixed, sub*spaces* of the set of all $\mathcal{H}$-matrices $\mathcal{M}_{\mathcal{H}}(I \times I, P_2, k)$. The second step is to restrict the spaces corresponding to the blocks $b = \tau \times \sigma \in P_2$ to tensor product form. The last step is to consider nested spaces.

3.1 Restriction to a Fixed Subspace

The set $\mathcal{M}_{\mathcal{H}}(I \times I, P_2, k)$ is, in general, not a vector space: Each block $b \in P_2$ is in the set

$$\mathcal{R}_b := \{A \in \mathbb{K}^b \ : \ \operatorname{rank}(A) \leq k_b\},$$

and this set is not a vector space, not even convex. This is a disadvantage considering the theoretical treatment of $\mathcal{H}$-matrices, since standard linear algebra cannot be applied.

In order to remedy this, we fix a k_b-dimensional subspace $\mathcal{V}_b$ of $\mathcal{R}_b$ for each $b \in P_2$ and define the space

$$\mathcal{M}_{\mathcal{H}}(I \times I, P_2, \mathcal{V}) := \{A \in \mathbb{K}^{I \times I} \ : \ A|_b \in \mathcal{V}_b\},$$

where $\mathcal{V} = (\mathcal{V}_b)_{b \in P_2}$.
If we equip $\mathbb{K}^{I \times I}$ with the Frobenius inner product

$$\langle A, B \rangle_F := \sum_{i,j \in I} a_{ij} \overline{b_{ij}},$$

we get a Hilbert space with the standard Frobenius norm of which $\mathcal{M}_{\mathcal{H}}(I \times I, P_2, \mathcal{V})$ is a subspace.

So the best approximation (with respect to the Frobenius norm) for any matrix A in this subspace can be expressed by means of the orthogonal projection

$$\Pi : \mathbb{K}^{I \times I} \to \mathcal{M}_{\mathcal{H}}(I \times I, P_2, \mathcal{V}) \quad \text{with}$$
$$\langle \Pi A, B \rangle_F = \langle A, B \rangle_F \quad \text{for all } A \in \mathbb{K}^{I \times I} \text{ and } B \in \mathcal{M}_{\mathcal{H}}(I \times I, P_2, \mathcal{V}).$$

3.2 Uniform $\mathcal{H}$-Matrices

In many practical applications, a certain $\tau \in T(I)$ occurs more than once in the blocks $b \in P_2$ of the given partition. Therefore, it is desirable to do as much work on the "cluster level" and as little as possible on the "block level" in order to reduce the total amount of work.

One way of doing this is to fix spaces $\mathcal{V}_{r,\tau}$ (corresponding to rows of the matrix) and $\mathcal{V}_{c,\tau}$ (corresponding to columns), both with dimension k_τ, and choosing $\mathcal{V}_b$ for $b = \tau \times \sigma \in P_2$ to be

$$\mathcal{V}_b := \operatorname{span}\{u_\tau v_\sigma^H \ : \ u_\tau \in \mathcal{V}_{r,\tau}, v_\sigma \in \mathcal{V}_{c,\sigma}\},$$

i.e., the tensor product of $\mathcal{V}_{r,\sigma}$ and $\mathcal{V}_{c,\tau}$. As usual for tensor product spaces, the dimension of $\mathcal{V}_b$ is $k_\tau k_\sigma$. The rank of matrices in this space is obviously bounded by $k_b = \min\{k_\tau, k_\sigma\}$.

If for each $b \in P_2$ the space $\mathcal{V}_b$ is defined as above, $\mathcal{M}_{\mathcal{H}}(I \times I, P_2, \mathcal{V})$ is denoted by $\mathcal{M}_{\mathcal{H}}(I \times I, P_2, \mathcal{V}_c, \mathcal{V}_r)$ and called the space of *uniform* $\mathcal{H}$-matrices with respect to I, P_2, $\mathcal{V}_c = (\mathcal{V}_{c,\tau})_{\tau \in T(I)}$, and $\mathcal{V}_r = (\mathcal{V}_{r,\tau})_{\tau \in T(I)}$.

In standard applications, the spaces $\mathcal{V}_{r,\tau}$ and $\mathcal{V}_{c,\sigma}$ will coincide for $\tau = \sigma$, but this is not required.

If we choose bases for $\mathcal{V}_{r,\tau}$ and $\mathcal{V}_{c,\tau}$, i.e., matrices $V_{r,\tau} \in \mathbb{K}^{\tau \times k_\tau}$ and $V_{c,\tau} \in \mathbb{K}^{\tau \times k_\tau}$ of full rank satisfying

$$\mathcal{V}_{r,\tau} = \operatorname{range} V_{r,\tau} \quad \text{and} \quad \mathcal{V}_{c,\tau} = \operatorname{range} V_{c,\tau},$$

we get the characterisation

$$\mathcal{V}_b = \{V_{r,\tau} S_b V_{c,\sigma}^H \ : \ S_b \in \mathbb{K}^{k_\tau \times k_\sigma}\}. \tag{3.1}$$

The tensor product structure can be used in the following way to reduce the total amount of work per matrix-vector multiplication:

The computation of a matrix-vector product $y_\tau = M_b x_\sigma$ with $M_b \in \mathcal{V}_b$ is equivalent to the computation of $y_\tau = V_{r,\tau} S_b V_{c,\sigma}^H x_\sigma$. The three matrix-vector products occurring in this expression can be used to devise an algorithm consisting of three steps for computing matrix-vector products with uniform $\mathcal{H}$-matrices:

First Step: Forward Transformation Since $\hat{x}_\sigma := V_{c,\sigma}^H x_\sigma$ does not depend on b, but only on σ, it has to be computed only once for each cluster. The procedure of computing all $\hat{x}_\sigma$ is called *forward transformation*:

$$\texttt{for } \sigma \in T(I) \texttt{ do } \hat{x}_\sigma := V_{c,\sigma}^H x|_\sigma$$

Second Step: Multiplication The multiplication with S_b has to be done for each block $b \in P_2$, but it has only a complexity of $\mathcal{O}(k_\tau k_\sigma)$, not of $\mathcal{O}((n_\tau + n_\sigma)k_b)$ as in the case of original $\mathcal{H}$-matrices:

$$\begin{array}{l}\texttt{for } \tau \in T(I) \texttt{ do } \hat{y}_\tau := 0;\\ \texttt{for } b = \tau \times \sigma \in P_2 \texttt{ do } \hat{y}_\tau := \hat{y}_\tau + S_b \hat{x}_\sigma\end{array}$$

Third Step: Backward Transformation The last step is the computation of y_τ by multiplying by $V_{r,\tau}$. As in the first step, this operation is performed only once per cluster:

$$\texttt{for } \tau \in T(I) \texttt{ do } y|_\tau := y|_\tau + V_{r,\tau} \hat{y}_\tau$$

3.3 Nested Spaces

Although the amount of work per matrix-vector multiplication can be significantly reduced by using uniform $\mathcal{H}$-matrices instead of full $\mathcal{H}$-matrices, it is, for standard applications with constant $k_\tau \equiv k$, still of order $\mathcal{O}(nk \log n)$, i.e., not optimal.

The logarithmic factor is caused by the forward and backward transformations, while the multiplication step typically has a complexity of $\mathcal{O}(nk)$. The basic idea for reaching the same complexity for the transformations is to reuse data: For a given level ℓ, $k2^\ell$ coefficients will be computed by the forward transformation. So there will be "more information" on finer levels (i.e., $T_\ell(I)$ with large ℓ, corresponding to small clusters) than on coarser levels, therefore there is a chance of devising a scheme that allows us to use the coefficients computed on lower levels in order to compute those on higher levels.

We make two observations:

1. If the coarser bases cannot be expressed in terms of the finer bases, the approach given above must fail since then information is lost when switching from finer to coarser levels.

2. In order to get a reasonable complexity, the computation of $\hat{x}_\tau$ for a given $\tau \in T(I) \setminus \mathcal{L}(T(I))$ should require no more than $\mathcal{O}(k_\tau^2)$ operations.

A simple way to meet these criteria is to require that the sets $\mathcal{V}_r$ and $\mathcal{V}_c$ form a nested hierarchy:

Definition 3.1 *A hierarchy $\mathcal{V} = (\mathcal{V}_\tau)_{\tau \in T(I)}$ of spaces is* consistent, *if for all $\tau \in T(I) \setminus \mathcal{L}(T(I))$ and all $\tau' \in S(\tau)$ we have*

$$\mathcal{V}_{r,\tau'} \supseteq \{u' = u|_{\tau'} \ : \ u \in \mathcal{V}_{r,\tau}\} \quad \text{and} \quad \mathcal{V}_{c,\tau'} \supseteq \{v' = v|_{\tau'} \ : \ v \in \mathcal{V}_{c,\tau}\}. \tag{3.2}$$

If, e.g., $\mathcal{V}_r$ is consistent, the space $\mathcal{V}_{r,\tau}$ for a given $\tau \in T(I) \setminus \mathcal{L}(T(I))$ is spanned by zero-extensions of vectors in the spaces $\{\mathcal{V}_{r,\tau'} \ : \ \tau' \in S(\tau)\}$ corresponding to its sons, so the coefficients $\hat{x}_\tau$ can be computed using the coefficients $\hat{x}_{\tau'}$ corresponding to the sons of τ, i.e., to the next-lower level.

Definition 3.2 ($\mathcal{H}^2$-matrix) *Let a partitioning P_2 and consistent families of spaces $\mathcal{V}_r = (\mathcal{V}_{r,\tau})_{\tau \in T(I)}$, $\mathcal{V}_c = (\mathcal{V}_{c,\tau})_{\tau \in T(I)}$ be given. Then the set of* $\mathcal{H}^2$-matrices *is given by*

$$\mathcal{M}_{\mathcal{H}^2}(I \times I, P_2, \mathcal{V}_r, \mathcal{V}_c) := \{M \in \mathbb{R}^{I \times I} \ : \ M|_b \in \mathcal{V}_b \text{ for all } b \in P_2\}.$$

An immediate conclusion from (3.2) is

Remark 3.3 *Let $b = \tau \times \sigma \in P_2$. The restriction of $M|_b \in \mathcal{V}_b$ to a subblock $b' = \tau' \times \sigma'$, $b' \in S(b)$, yields a matrix $M|_{b'} \in \mathcal{V}_{b'}$.*

The consistency condition can be reformulated for the matrices $V_{r,\tau}, V_{c,\sigma}$:

Lemma 3.4 *The families $\mathcal{V}_r$ and $\mathcal{V}_c$ with their corresponding bases $(V_{r,\tau})_{\tau \in T(I)}$ and $(V_{c,\tau})_{\tau \in T(I)}$ are consistent if and only if there exist matrices $B_{r,\tau',\tau}, B_{c,\tau',\tau} \in \mathbb{K}^{k_{\tau'} \times k_\tau}$ satisfying*

$$V_{r,\tau}|_{\tau' \times k_\tau} = V_{r,\tau'} B_{r,\tau',\tau} \quad \text{and} \quad V_{c,\tau}|_{\tau' \times k_\tau} = V_{c,\tau'} B_{c,\tau',\tau} \tag{3.3}$$

for all $\tau \in T(I) \setminus \mathcal{L}(T(I))$, $\tau' \in S(\tau)$.

Using the matrices $B_{c,\sigma',\sigma}$ from Lemma 3.4, we can express $\hat{x}_\sigma$ in terms of $\hat{x}_{\sigma'}$: Let $S(\sigma) = \{\sigma_1, \ldots, \sigma_s\}$ be the set of sons of σ. We find

$$V_{c,\sigma} = \begin{pmatrix} V_{c,\sigma_1} B_{c,\sigma_1,\sigma} \\ \vdots \\ V_{c,\sigma_s} B_{c,\sigma_s,\sigma} \end{pmatrix}, \tag{3.4}$$

leading to

$$\begin{aligned} \hat{x}_\sigma = V_{c,\sigma}^H x|_\sigma &= \begin{pmatrix} B_{c,\sigma_1,\sigma}^H V_{c,\sigma}^H & \ldots & B_{c,\sigma_s,\sigma}^H V_{c,\sigma}^H \end{pmatrix} \begin{pmatrix} x|_{\sigma_1} \\ \vdots \\ x|_{\sigma_s} \end{pmatrix} \\ &= \sum_{i=1}^{s} B_{c,\sigma_i,\sigma}^H V_{c,\sigma_i}^H x|_{\sigma_i} = \sum_{i=1}^{s} B_{c,\sigma_i,\sigma}^H \hat{x}_{\sigma_i}, \end{aligned} \tag{3.5}$$

i.e., the coefficient vector $\hat{x}_\sigma$ can be indeed computed using only the coefficient vectors $\hat{x}_{\sigma'}$ corresponding to its immediate sons. This leads to an improved version of the transformations introduced for uniform $\mathcal{H}$-matrices:

Fast Forward Transformation In order to make use of (3.5), we compute the coefficients $\hat{x}_\sigma$ proceeding from the "finer" towards the "coarser" levels:

```
procedure FastForward(σ);
begin
  if S(σ) = ∅ then x̂_σ := V^H_{c,σ} x|_σ
  else
    begin
      x̂_σ := 0;
      for σ' ∈ S(σ) do
        begin
          FastForward(σ'); x̂_σ := x̂_σ + B^H_{c,σ',σ} x̂_σ'
        end
    end
end
```

Fast Backward Transformation In a similar fashion, we can represent $\hat{y}_\tau$ in terms of the coefficients $\hat{y}_{\tau'}$ corresponding to the sons τ' of τ, so by proceeding from the "coarser" towards the "finer" levels and updating the coefficients according to (3.4), we get the following algorithm:

```
procedure FastBackward(τ);
begin
  if S(τ) = ∅ then y|_τ := V_{r,τ} ŷ_τ
  else
    for τ' ∈ S(τ) do
      begin
        ŷ_τ' := ŷ_τ' + B_{r,τ',τ} ŷ_τ; FastBackward(τ')
      end
end
```

In the case of constant k_τ, the "fast" versions of the transformations require only $\mathcal{O}(k^2)$ operations to compute the coefficients corresponding to a certain cluster.

Remark 3.5 (Relation to multilevel methods) *There are similarities between multilevel methods and $\mathcal{H}^2$-matrices. For the sake of simplicity, we assume that $T_{\ell_{\max}}(I) = \mathcal{L}(\mathcal{T}(I))$ holds, i.e., that all leaves of $\mathcal{T}(I)$ occur on the finest level $\ell_{\max}$.*

We define the spaces

$$\mathcal{V}_\ell := \{u \in \mathbb{K}^I \ : \ u|_\tau \in \mathcal{V}_{c,\tau} \text{ for all } \tau \in T_\ell(I)\}$$

for each $\ell \in \{0, \ldots, \ell_{\max}\}$. They form a hierarchy of nested subspaces of $\mathbb{K}^I$, similar to the hierarchies of finite element spaces used in the context of multilevel methods.

As in finite element methods, a function $y \in \mathcal{V}_\ell$ can be represented by a vector $(\hat{y}_\tau)_{\tau \in T_\ell(I)}$ of coefficients in the bases $(\mathcal{V}_{c,\tau})_{\tau \in T_\ell(I)}$.

The inner loop of the fast backward transformation resembles the prolongation operator of standard multilevel methods: If we start with a coefficient vector $(\hat{y}_\tau)_{\tau\in T_\ell(I)}$ corresponding to a coarse grid, it will compute a coefficient vector $(B_{\tau,\tau'},\tau\hat{y}_\tau)_{\tau'\in T_{\ell+1}(I)}$ corresponding to the next finer grid.

The outer loop of the fast backward transformation and the corresponding accumulation across all levels is an operation frequently used in the context of additive multilevel methods.

Since the fast forward transformation is simply the adjoint of the fast backward transformation, it corresponds to the restriction operator of standard multilevel methods: A function, given on the "finest" level, is transferred to "coarser" levels.

3.4 Choice of k_τ

The simplest choice of the dimension k_τ is a constant k_{const}. Since the dimension cannot exceed the size n_τ of a cluster, the exact requirement is

$$k_\tau = \min\{k_{\text{const}}, n_\tau\}. \tag{3.6}$$

In the case of the example from Subsection 2.5, k_τ depends only on the cluster level:

$$k_\tau = k_\ell := \min\{k_{\text{const}}, 2^{p-\ell}\}$$

for all $\tau \in T_\ell(I)$.

If an $\mathcal{H}^2$-matrix has to approximate a BEM matrix up to the error $\mathcal{O}(h^\gamma)$ with γ being the consistency order, the choice of k_{const} should be of the order $\log n$, where $n := \#I$.

As proved in [6], it is not necessary to use $k_{\text{const}} = \log n$ for all blocks. This size is only needed for large blocks. A reasonable choice is

$$k_\tau := \ell_{\max} - \text{level}(\tau) + 1$$

or, more generally,

$$k_\tau := \min\{n_\tau, \alpha + \beta(\ell_{\max} - \text{level}(\tau))\} \qquad \text{for some } \alpha, \beta \geq 1. \tag{3.7}$$

If n_τ depends only on $\text{level}(\tau)$, also $k_\tau = k_\ell$ is only a function of the level number. Note that for small-sized clusters, $k(\tau)$ is of order 1, while large clusters (with small $\text{level}(\tau)$) lead to $k_\tau = \mathcal{O}(\log n)$, since $\ell_{\max} = \mathcal{O}(\log n)$.

3.5 Choice of the Bases in the Case of Piecewise Constant Functions

We consider the one-dimensional case of piecewise constant functions on X_i, $i \in I$, where X_i is an interval. We choose k_τ to be "constant" according to (3.6).

Given an integral operator with a kernel function $\kappa(x, y)$, consider the expansion

$$\kappa(x,y) = \sum_{\nu,\mu=0}^{\infty} s_{\nu,\mu} P_\nu^\tau(x) P_\mu^\sigma(y) \qquad \text{in the range of } (x,y) \in X_\tau \times X_\sigma \tag{3.8}$$

with X_τ, X_σ (cf. (2.5)) satisfying the admissibility condition (2.6). Here, $\{P_\mu^\sigma : \mu \in \mathbb{N}_0\}$ is the orthogonal system of Legendre polynomials; more precisely, P_μ^σ is the μth Legendre polynomial transformed from $[-1,1]$ onto the interval X_σ such that

$$\langle P_\nu^\sigma, P_\mu^\sigma \rangle_{L^2(X_\sigma)} = \delta_{\nu,\mu} \text{ (Kronecker symbol)}.$$

We consider a finite part of the expansion (3.8) by setting $K_\tau := \{0, \ldots, k_\tau - 1\}$ and

$$\kappa_{k_\tau,k_\sigma}(x,y) := \sum_{\nu \in K_\tau} \sum_{\mu \in K_\sigma} s_{\nu,\mu} P_\nu^\tau(x) P_\mu^\sigma(y). \tag{3.9}$$

In the Galerkin setting, the entry a_{ij} of the discrete matrix A is given by the double integral

$$a_{ij} = \int_{X_i} \int_{X_j} \kappa(x,y) b_i(x) b_j(y) dx dy$$

(we recall that $X_i = \operatorname{supp} b_i$ and $X_j = \operatorname{supp} b_j$).

We replace κ by the approximation κ_{k_τ,k_σ} and get

$$\begin{aligned} a_{ij} &\approx \int_{X_i} \int_{X_j} \kappa_{k_\tau,k_\sigma}(x,y) b_i(x) b_j(y) dx dy \\ &= \sum_{\nu \in K_\tau} \sum_{\mu \in K_\sigma} s_{\nu,\mu} \int_{X_i} P_\nu^\tau(x) b_i(x) dx \int_{X_j} P_\mu^\sigma(y) b_j(y) dy \\ &= \sum_{\nu \in K_\tau} \sum_{\mu \in K_\sigma} s_{\nu,\mu} \int_{X_\tau} P_\nu^\tau(x) b_i(x) dx \int_{X_\sigma} P_\mu^\sigma(y) b_j(y) dy, \end{aligned}$$

so by setting

$$\begin{aligned} \tilde{V}_{r,\tau} &:= \left(\int_{X_\tau} P_\nu^\tau(x) b_i(x) dx \right)_{i \in \tau, \nu \in K} \in \mathbb{R}^{\tau \times k_\tau}, \\ \tilde{V}_{c,\sigma} &:= \left(\int_{X_\sigma} P_\mu^\sigma(y) b_j(y) dy \right)_{j \in \sigma, \mu \in K} \in \mathbb{R}^{\sigma \times k_\sigma}, \end{aligned}$$

we have

$$a_{ij} \approx \sum_{\nu=0}^{k_\tau - 1} \sum_{\mu=0}^{k_\sigma - 1} s_{\nu,\mu} (V_{r,\tau})_{i,\nu} (V_{c,\sigma})_{j,\mu} = (\tilde{V}_{a,\tau} S \tilde{V}_{c,\sigma}^T)_{ij} \quad \text{for } i \in \tau \text{ and } j \in \sigma$$

if we set $S = (s_{\nu,\mu})_{\nu,\mu} \in \mathbb{R}^{k_\tau \times k_\sigma}$. We let

$$\mathcal{V}_{r,\tau} = \operatorname{range} \tilde{V}_{r,\tau} \quad \text{and} \quad \mathcal{V}_{c,\sigma} = \operatorname{range} \tilde{V}_{c,\sigma},$$

and find that the approximation $\tilde{V}_{r,\tau} S \tilde{V}_{c,\sigma}^T$ is in $\mathcal{V}_b$ (cf. (3.1)) for $b = \tau \times \sigma$. The dimension of $\mathcal{V}_{r,\tau}$ (or $\mathcal{V}_{c,\sigma}$) is k_{const} for $n_\tau \geq k_{\text{const}}$ (or $n_\sigma \geq k_{\text{const}}$) and n_τ (or n_σ) otherwise, satisfying (3.6).

Now we consider the consistency condition (3.2) in the case of "constant" k according to (3.6). Let $\tau' \in S(\tau)$.

For each polynomial P_ν^τ of degree $< k_{\text{const}}$, the restriction $P_\nu^\tau|_{X_{\tau'}}$ is again a polynomial of degree $< k_{\text{const}}$. Therefore, there are coefficients $b_{\nu,\nu'}^{\tau,\tau'}$ satisfying

$$P_\nu^\tau|_{X_{\tau'}} = \sum_{\nu' \in K_{\tau'}} b_{\nu,\nu'}^{\tau,\tau'} P_{\nu'}^{\tau'}|_{X_{\tau'}}.$$

We have

$$\begin{aligned}\tilde{V}_{r,\tau}|_{\tau' \times k_\tau} &= \left(\int_{X_{\tau'}} P_\nu^\tau(x) b_i(x) dx \right)_{i \in \tau', \nu} = \left(\int_{X_{\tau'}} \sum_{\nu' \in K_{\tau'}} b_{\nu,\nu'}^{\tau,\tau'} P_{\nu'}^{\tau'}(x) b_i(x) dx \right)_{i,\nu} \\ &= \left(\sum_{\nu' \in K_{\tau'}} b_{\nu,\nu'}^{\tau,\tau'} \int_{X_{\tau'}} P_{\nu'}^{\tau'}(x) b_i(x) dx \right)_{i,\nu} = \tilde{V}_{r,\tau'} \underbrace{\left(b_{\nu,\nu'}^{\tau,\tau'} \right)_{\nu,\nu'}}_{=:B_{a,\tau,\tau'}},\end{aligned}$$

so

$$\mathcal{V}_{r,\tau}|_{\tau'} = \operatorname{range} \tilde{V}_{r,\tau}|_{\tau' \times k_\tau} = \operatorname{range} \tilde{V}_{r,\tau'} B_{r,\tau,\tau'} \subseteq \mathcal{V}_{r,\tau'} \tag{3.10}$$

holds, therefore the condition (3.2) is fulfilled.

Remark 3.6 *In the case of variable k (cf. (3.7)), this simple polynomial-based construction is no longer possible: The rank of a polynomial on a cluster τ may be larger than that of all the polynomials on one of its sons $\tau' \in S(\tau)$, so the restriction property of the spaces cannot be derived from the restriction property of the polynomials.*

In [9], the polynomials are replaced by other functions that satisfy the restriction property while still allowing a suitable approximation of certain kernel functions.

4 Complexity and Storage Requirements

4.1 Storage Requirements

Assume the cluster tree from Subsection 2.5.

Remark 4.1 *If k_τ is "constant" (cf. (3.6)), the restriction matrices $B_{r,\tau',\tau}, B_{c,\tau',\tau}$ require not more than $\mathcal{O}(nk_{\text{const}})$ storage.*

Proof. The number of clusters on level ℓ is bounded by $\#T_\ell(I) \leq 2^\ell$. We have to store the $k_{\tau'} \times k_\tau$-matrices $B_{r,\tau',\tau}$ and $B_{c,\tau',\tau}$ for each father-son pair $\tau \in T_\ell(I), \tau' \in S(\tau)$, so the total storage requirement is

$$S_{\text{const}} := 2 \sum_{\ell=0}^{\ell_{\max}-1} \sum_{\tau \in T_\ell(I)} \sum_{\tau' \in S(\tau)} k_\tau k_{\tau'}.$$

Let $\#I = n = 2^p$ and set

$$\ell_c := \min\{\ell \ : \ \forall \tau \in T_\ell(I) \ : \ n_\tau \leq k_{\text{const}}\} = \lfloor p - \log_2(k_{\text{const}})\rfloor. \tag{4.1}$$

Then we can split the sum as follows:

$$\begin{aligned}
S_{\text{const}} &= 2\sum_{\ell=0}^{\ell_c-1}\sum_{\tau\in T_\ell(I)}\sum_{\tau'\in S(\tau)} k_\tau k_{\tau'} + 2\sum_{\ell=\ell_c}^{\ell_{\max}}\sum_{\tau\in T_\ell(I)}\sum_{\tau'\in S(\tau)} k_\tau k_{\tau'} \\
&\leq 2\sum_{\ell=0}^{\ell_c-1} 2\#T_\ell(I)k_{\text{const}}^2 + 2\sum_{\ell=\ell_c}^{\ell_{\max}}\sum_{\tau\in T_\ell(I)}\sum_{\tau'\in S(\tau)} n_\tau n_{\tau'} \\
&\leq 4\sum_{\ell=0}^{\ell_c-1} 2^\ell k_{\text{const}}^2 + 2\sum_{\ell=\ell_c}^{\ell_{\max}}\sum_{\tau\in T_\ell(I)} (n_\tau)^2 \leq 4k_{\text{const}}^2 2^{\ell_c} + 2\sum_{\ell=\ell_c}^{\ell_{\max}} 2^\ell 2^{2p-2\ell} \\
&\leq 4k_{\text{const}}^2 \frac{2\ 2^p}{k_{\text{const}}} + 2\ 2^{2p}\sum_{\ell=\ell_c}^{\ell_{\max}} 2^{-\ell} \leq 8k_{\text{const}}n + 4\ 2^{2p}\ 2^{-\ell_c} \\
&\leq 8k_{\text{const}}n + 4\ 2^{2p}\ \frac{k_{\text{const}}}{2^p} = 12k_{\text{const}}n.
\end{aligned}$$

■

In [6], a proof of the following estimate for the case of variable k_τ can be found:

Remark 4.2 *If k_τ grows linearly (cf. (3.7)), the restriction matrices $B_{r,\tau^\tau,\tau}, B_{c,\tau',\tau}$ require not more than $\mathcal{O}(n)$ storage.*

Note that the matrices $B_{r,\tau',\tau}$ and $B_{c,\tau',\tau}$ describing the $\mathcal{H}^2$-format are to be computed and stored only once for a subspace of $\mathcal{H}^2$-Matrices, since they are independent of any individual $\mathcal{H}^2$-matrix.

Each matrix $M \in \mathcal{M}_{\mathcal{H}^2}(I \times I, P_2, \mathcal{V}_a, \mathcal{V}_c)$ is completely described by the matrices $(S_b)_{b\in P_2}$ from (3.1) carrying the entries $s_{\nu,\mu}$ from (3.9).

Using a similar proof as above, the following estimate can be derived:

Remark 4.3 *The storage needed for all block coefficient matrices $(S_b)_{b\in P_2}$ is bounded by $\mathcal{O}(n)$ in the variable case (3.7) and by $\mathcal{O}(nk_{\text{const}})$ in the constant case (3.6).*

4.2 Complexity Bound for Arithmetical Operations

4.2.1 Fast Matrix-Vector Multiplication

As mentioned in Subsections 3.2 and 3.3, the fast matrix-vector multiplication algorithm is performed in three steps: (i) a *forward transformation* computes the coefficients of a given vector in the bases corresponding to each cluster, (ii) in a *block-multiplication* phase, the "scaling matrices" S_b are applied to these coefficient before the (iii) *backward transformation* computes the result vector from the coefficients on all clusters.

It can be easily seen that in all the steps the arithmetical work is proportional to the number of entries in the matrices $B_{r,\tau',\tau}, B_{c,\tau',\tau}$ and S_b (cf. [6]). Therefore, the previous results about the storage show a respective complexity of $\mathcal{O}(n)$ or $\mathcal{O}(nk_{\text{const}})$.

4.2.2 Matrix Addition with matching Bases

Different from general $\mathcal{H}$-matrices, the sum of two $\mathcal{H}^2$-matrices (with the same partitioning and the same nested bases) can be performed *exactly*. Since only the matrices $(S_b)_{b\in P_2}$ are to be added, the cost is clearly the same as the storage needed for the family $(S_b)_{b\in P_2}$.

4.2.3 Matrix Addition with arbitrary Bases

The sum of two $\mathcal{H}^2$-matrices A and B with different bases but identical P_2 and $\mathcal{T}(I)$ can be represented *exactly* by doubling the dimensions k_τ and simply copying the data associated with A and B, so this operation requires $\mathcal{O}(n)$ operations in the case of variable rank and $\mathcal{O}(nk_{\text{const}})$ in the case of constant rank.

Of course, the potential doubling of the ranks leads to an unattractive complexity when computing sums of a large number of matrices. To avoid this, the algorithm introduced in Section 5 can be applied after each addition to reduce the dimensions by removing unnecessary basis functions.

5 Approximation of General Matrices by $\mathcal{H}^2$-Matrices

The *a priori* computation of the bases for an $\mathcal{H}^2$-Matrix requires that the kernel κ and its expansion (3.8) are known. In many applications, this requirement is not met, so another method of computing these bases is desirable.

In this section, we present an algorithm that computes an approximation of the optimal nested bases from a given general matrix. Furthermore, a certain accuracy can be guaranteed by applying a slight modification of the algorithm that will be discussed in Section 7.

We assume that a suitable partitioning P_2 is given. Since we allow *general* matrices with n^2 entries in the first stage, the following algorithm needs at least $\mathcal{O}(n^2)$ operations.

5.1 Optimal Approximation on One Level

We fix a $\tau \in T_{\ell_{\max}-1}$ and want to compute an optimal $\mathcal{V}_{r,\tau}$. Before we can do this, we first have to investigate the meaning of "optimal" in this context. The space $V_{r,\tau}$ will be used in all blocks $b \in P_2$ that have the form $b = \tau \times \sigma$ for a certain $\sigma \in T(I)$, so it seems straightforward to minimise

$$\sum_{b\in P_\tau} \|A|_b - \Pi_{\mathcal{V}_{r,\tau}} A|_b\|_F^2 \tag{5.1}$$

with respect to all $\mathcal{V}_{r,\tau}$, where $\Pi_{\mathcal{V}_\tau,a}$ is the orthogonal projection from $\mathbb{R}^\tau$ to $\mathcal{V}_{\tau,a}$ and P_τ is defined by

$$P_\tau := \{b \in P_2 \ : \ \exists \sigma \in T(I) : b = \tau \times \sigma\} = \{\tau \times \sigma \ : \ \sigma \in \operatorname{cols}(\tau, P_2)\}. \tag{5.2}$$

This minimisation will *not* lead to a good approximation of A, since the space $\mathcal{V}_{r,\tau}$ is not only used for $b \in P_\tau$, but, due to the consistency condition (3.2), acts as a restriction for the computation of $\mathcal{V}_{r,\tau'}$ for all $\tau' \in T(I)$ with $\tau \subseteq \tau'$.

So we aim to minimise the quantity

$$\sum_{b \in P_\tau^+} \|A_b - \Pi_{\mathcal{V}_{r,\tau}} A_b\|_F^2, \tag{5.3}$$

summed over all blocks from

$$P_\tau^+ := \{b \ : \ \exists \sigma \in T(I) : b = \tau \times \sigma \text{ and } \exists b^+ \in P_2 : b = b^+ \cap \tau \times I\}, \tag{5.4}$$

i.e., all blocks b that are subsets of a block $b^+ \in P_2$ and that have the form $b = \tau \times \sigma$ for a $\sigma \in T(I)$.

Note that the sets $b \in P_\tau^+$ are disjoint because P_2 is a partitioning of disjoint blocks.

Since all elements of P_τ^+ have the form $\tau \times \sigma$, we can introduce the sets of clusters and indices corresponding to blocks in P_τ^+:

$$C_\tau := \{\sigma \in T(I) \ : \ \tau \times \sigma \in P_\tau^+\} \quad \text{and} \quad I_\tau := \bigcup C_\tau.$$

Obviously, we have $P_\tau^+ = \{\tau \times \sigma \ : \ \sigma \in C_\tau\}$, so the expression (5.3) can be rewritten as

$$\sum_{b \in P_\tau^+} \|A_b - \Pi_{\mathcal{V}_{r,\tau}} A_b\|_F^2 = \|A_\tau - \Pi_{\mathcal{V}_{r,\tau}} A_\tau\|_F^2 \tag{5.5}$$

with

$$A_\tau := (\chi_{C_\tau}(\sigma) A_{\tau \times \sigma})_{\sigma \in T(I)} \in \mathbb{R}^{\tau \times I}, \tag{5.6}$$

where χ_{C_τ} is the characteristic function of C_τ, i.e.,

$$\chi_{C_\tau}(\sigma) = \begin{cases} 1 & \text{if } \sigma \in C_\tau \\ 0 & \text{else} \end{cases}.$$

In Figure 2, the blocks in P_τ^+ for a given partitioning P_2 corresponding to different clusters τ are marked.

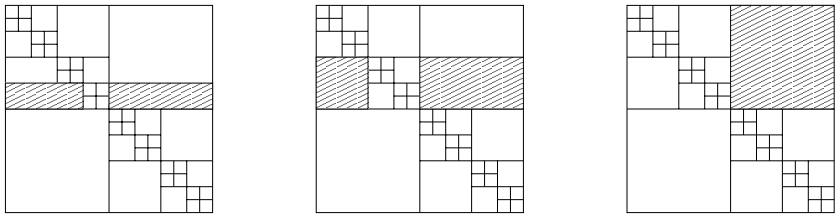

Figure 2: Blocks P_τ^+ for different clusters

It is well-known that approximation problems of this type can be handled by using the singular value decomposition of A_τ: We let $\mathrm{id}_{p\times q} := (\delta_{ij})_{i,j=1}^{p,q}$ for all $p, q \in \mathbb{N}$. Let $l \in \mathbb{N}_{\geq k_\tau}$ be the rank of A_τ. Then there are $U \in \mathbb{K}^{\tau\times l}, V \in \mathbb{K}^{I_\tau\times l}$ and a matrix $\Sigma := \mathrm{diag}(\sigma_1, \ldots, \sigma_l)$ with

$$A_\tau = U\Sigma V^H, \quad \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_l > 0, \quad U^H U = V^H V = \mathrm{id}_{l\times l}.$$

To determine the optimal rank-k-approximation of A_τ, we replace Σ by the truncated diagonal matrix $\tilde{\Sigma} := \mathrm{diag}(\sigma_1, \ldots, \sigma_{k_\tau}, 0, \ldots, 0) \in \mathbb{R}^{l\times l}$ and set $\tilde{A}_\tau = U\tilde{\Sigma}V^H$. We still have to show that $\tilde{A}_\tau$ is of the form $\Pi_{\mathcal{V}_{r,\tau}} A_\tau$ for a k_τ-dimensional subspace $\mathcal{V}_{r,\tau}$ of $\mathbb{R}^\tau$.

We let $\tilde{U} := U\,\mathrm{id}_{l,k_\tau}$, i.e., $\tilde{U}$ contains the first k_τ columns of U. Since $\tilde{U}^H\tilde{U} = \mathrm{id}_{k_\tau\times k_\tau}$, the mapping $\Pi_{\tilde{U}} := \tilde{U}\tilde{U}^H$ is the orthogonal projection onto $\mathcal{V}_{r,\tau} := \mathrm{range}(\tilde{U})$ satisfying

$$\begin{aligned}\Pi_{\tilde{U}} A_\tau &= \tilde{U}\tilde{U}^H U\Sigma V^H = U\,\mathrm{id}_{l\times k_\tau}\,\mathrm{id}_{l\times k_\tau}^H\, U^H U\Sigma V^H \\ &= U\,\mathrm{id}_{l\times k_\tau}\,\mathrm{id}_{l\times k_\tau}^H\,\Sigma V^H = U\tilde{\Sigma}V^H = \tilde{A}_\tau.\end{aligned}$$

Since $\tilde{A}_\tau$ is the optimal rank-k-approximation of A_τ, $\mathcal{V}_{r,\tau} = \mathrm{range}(\tilde{U})$ is the optimal choice.

Since we are only interested in $\mathcal{V}_{r,\tau}$, we can eliminate the computation of the matrix V by characterising U and Σ in terms of the Gram matrix

$$G_\tau := A_\tau A_\tau^H = U\Sigma\Sigma^H U^H = U\Sigma^2 U^H$$

and thus arriving at a symmetric positive semidefinite eigenvalue problem.

Remark 5.1 (Adaptivity) *The approximation error is given by*

$$\|A_\tau - \Pi_{\mathcal{V}_{r,\tau}} A_\tau\|_F^2 = \sum_{\nu=k_\tau+1}^{l} \sigma_\nu^2,$$

so it is possible to choose k_τ adaptively in order to ensure that the approximation error is below a given bound.

Applying this method to A^H instead of A, we can find the optimal column-related spaces $\mathcal{V}_{c,\tau}$.

5.2 Global Approximation by a Greedy-Type Algorithm

Using the above procedure, we can compute optimal bases for all clusters, but they do not necessarily satisfy the consistency condition (3.2). In order to compute consistent bases, we start from the finest level $\ell_{\max}$.

We compute bases $V_{r,\tau'}$ for all clusters $\tau' \in T_{\ell_{\max}}(I)$ using the method described in Subsection 5.1. Let us now consider a cluster $\tau \in T_\ell(I)$ on the next level $\ell = \ell_{\max} - 1$. We want to compute a basis $V_{r,\tau}$ of a space $\mathcal{V}_{r,\tau}$ minimising

$$\|A_\tau - \Pi_{\mathcal{V}_{r,\tau}} A_\tau\|_F^2. \tag{5.7}$$

Since $\langle A_\tau - \Pi_{\mathcal{V}_{r,\tau}} A_\tau, \Pi_{\mathcal{V}_{r,\tau}} A_\tau \rangle_F = 0$, we get by the Pythagoras identity

$$\|A_\tau - \Pi_{\mathcal{V}_{r,\tau}} A_\tau\|_F^2 = \|A_\tau\|_F^2 - \|\Pi_{\mathcal{V}_{r,\tau}} A_\tau\|_F^2.$$

If we require the matrix $V_{r,\tau}$ to be orthonormal, we have $\Pi_{\mathcal{V}_{r,\tau}} = V_{r,\tau} V_{r,\tau}^H$ and find

$$\|\Pi_{\mathcal{V}_{r,\tau}} A_\tau\|_F^2 = \|V_{r,\tau} V_{r,\tau}^H A_\tau\|_F^2 = \|V_{r,\tau}^H A_\tau\|_F^2.$$

Therefore, the problem of minimising the error (5.7) can be reformulated as the problem of finding an orthonormal matrix $V_{r,\tau} \in \mathbb{K}^{\tau \times k_\tau}$ maximising

$$\|V_{r,\tau}^H A_\tau\|_F^2. \tag{5.8}$$

We do not only look for an optimal basis, but for a hierarchy of bases being *consistent*, so we have to satisfy the condition (3.3). Therefore, we reformulate our maximisation problem in terms of $B_{r,\tau',\tau}$: We look for a family $(B_{r,\tau',\tau})_{\tau' \in S(\tau)}$ of matrices maximising

$$\left\| \begin{pmatrix} B_{r,\tau_1,\tau} V_{r,\tau_1}^H & \dots & B_{r,\tau_s,\tau} V_{r,\tau_s}^H \end{pmatrix} A_\tau \right\|_F^2 = \left\| \begin{pmatrix} B_{r,\tau_1,\tau} \\ \vdots \\ B_{r,\tau_s,\tau} \end{pmatrix}^H \begin{pmatrix} V_{r,\tau_1}^H A_\tau|_{\tau_1 \times I} \\ \vdots \\ V_{r,\tau_s}^H A_\tau|_{\tau_s \times I} \end{pmatrix} \right\|_F^2$$

for $S(\tau) = \{\tau_1, \dots, \tau_s\}$.

Since $V_{r,\tau}$ has to be orthonormal and

$$V_{r,\tau} = \begin{pmatrix} V_{r,\tau_1} & & \\ & \ddots & \\ & & V_{r,\tau_s} \end{pmatrix} \begin{pmatrix} B_{r,\tau_1,\tau} \\ \vdots \\ B_{r,\tau_s,\tau} \end{pmatrix},$$

we get

$$I = V_{r,\tau}^H V_{r,\tau} = \begin{pmatrix} B_{r,\tau_1,\tau} \\ \vdots \\ B_{r,\tau_s,\tau} \end{pmatrix}^H \begin{pmatrix} B_{r,\tau_1,\tau} \\ \vdots \\ B_{r,\tau_s,\tau} \end{pmatrix},$$

so the combined matrix

$$Q_\tau := \begin{pmatrix} B_{r,\tau_1,\tau} \\ \vdots \\ B_{r,\tau_s,\tau} \end{pmatrix} \in \mathbb{K}^{m_\tau \times k_\tau} \tag{5.9}$$

with $m_\tau := \sum_{\tau' \in S(\tau)} k_{\tau'}$ must be orthonormal, too. We set

$$\widehat{A}_{\tau,\tau'} := V_{r,\tau'}^H A_\tau|_{\tau' \times I} \tag{5.10}$$

and

$$\bar{A}_\tau := \begin{pmatrix} \widehat{A}_{\tau,\tau_1} \\ \vdots \\ \widehat{A}_{\tau,\tau_s} \end{pmatrix}.$$

So now we look for an orthonormal matrix $Q_\tau \in \mathbb{K}^{m_\tau \times k_\tau}$ that maximises

$$\|Q_\tau^H \bar{A}_\tau\|_F^2.$$

This problem has the same structure as the original problem (5.8) and can therefore be treated in a similar fashion, i.e., by computing the eigenvectors of

$$\widehat{G}_\tau := \bar{A}_\tau \bar{A}_\tau^H = \begin{pmatrix} \widehat{A}_{\tau,\tau_1}\widehat{A}_{\tau,\tau_1}^H & \dots & \widehat{A}_{\tau,\tau_1}\widehat{A}_{\tau,\tau_s}^H \\ \vdots & \ddots & \vdots \\ \widehat{A}_{\tau,\tau_s}\widehat{A}_{\tau,\tau_1}^H & \dots & \widehat{A}_{\tau,\tau_s}\widehat{A}_{\tau,\tau_s}^H \end{pmatrix}. \tag{5.11}$$

Making use of (5.10) in combination with (5.6), we have

$$\begin{aligned} \widehat{A}_{\tau,\tau_i}\widehat{A}_{\tau,\tau_j}^H &= V_{r,\tau_i}^H A_\tau|_{\tau_i \times I} A_\tau|_{\tau_j \times I}^H V_{r,\tau_j} = \sum_{\sigma \in T(I)} V_{r,\tau_i}^H A_\tau|_{\tau_i \times \sigma} A_\tau|_{\tau_j \times \sigma} V_{r,\tau_j} \\ &= \sum_{\sigma \in T(I)} \chi_{C_\tau}(\sigma) V_{r,\tau_i}^H A_\tau|_{\tau_i \times \sigma} A_\tau|_{\tau_j \times \sigma} V_{r,\tau_j} = \sum_{\sigma \in C_\tau} V_{r,\tau_i}^H A|_{\tau_i \times \sigma} A|_{\tau_j \times \sigma}^H V_{r,\tau_j}, \end{aligned}$$

so we can rewrite the computation of $\widehat{G}_\tau$ in terms of single blocks by using

$$\widehat{G}_\tau = \sum_{\sigma \in C_\tau} \widehat{G}_{\tau,\sigma} \tag{5.12}$$

with $\widehat{G}_{\tau,\sigma} := \bar{A}_{\tau,\sigma}\bar{A}_{\tau,\sigma}^H$ for

$$\bar{A}_{\tau,\sigma} := \begin{pmatrix} \widehat{A}_{\tau_1,\sigma} \\ \vdots \\ \widehat{A}_{\tau_s,\sigma} \end{pmatrix} \tag{5.13}$$

with $\widehat{A}_{\tau',\sigma} := V_{r,\tau'}^H A|_{\tau' \times \sigma}$ for all $\tau' \in S(\tau)$.

The explicit computation of $\widehat{A}_{\tau',\sigma}$ takes $\mathcal{O}(k_{\tau'} n_{\tau'} n_\sigma)$ operations. Once more, it is possible to make use of the consistency condition in the form of equation (3.5) to get the equation

$$\widehat{A}_{\tau,\sigma} = V_{r,\tau}^H A|_{\tau \times \sigma} = \begin{pmatrix} V_{r,\tau_1} B_{r,\tau_1,\tau} \\ \vdots \\ V_{r,\tau_s} B_{r,\tau_s,\tau} \end{pmatrix}^H A|_{\tau \times \sigma} = \begin{pmatrix} V_{r,\tau_1} B_{r,\tau_1,\tau} \\ \vdots \\ V_{r,\tau_s} B_{r,\tau_s,\tau} \end{pmatrix}^H \begin{pmatrix} A|_{\tau_1 \times \sigma} \\ \vdots \\ A|_{\tau_s \times \sigma} \end{pmatrix}$$

$$= \begin{pmatrix} B_{r,\tau_1,\tau} \\ \vdots \\ B_{r,\tau_s,\tau} \end{pmatrix}^H \begin{pmatrix} V_{r,\tau_1}^H A|_{\tau_1\times\sigma} \\ \vdots \\ V_{r,\tau_s}^H A|_{\tau_s\times\sigma} \end{pmatrix} = Q_\tau^H \begin{pmatrix} \widehat{A}_{\tau_1,\sigma} \\ \vdots \\ \widehat{A}_{\tau_s,\sigma} \end{pmatrix} = Q_\tau^H \bar{A}_{\tau,\sigma},$$

i.e., by using the matrices $\widehat{A}_{\tau',\sigma}$ corresponding to the sons τ' of τ, we can compute $\widehat{A}_{\tau,\sigma}$ with only $\mathcal{O}(k_\tau m_\tau n_\sigma)$ operations.

Combining all the basic steps mentioned above with a recursive bottom-up strategy, we get the following algorithm:

```
procedure ComputeRowBasis(τ);
begin
  if S(τ) = ∅ then
    begin
      G_τ := 0;
      for σ ∈ C_τ do G_τ := G_τ + A|_{τ×σ} A|_{τ×σ}^H;
      Compute the Schur decomposition Q̃_τ^H G_τ Q̃_τ = D
        with D = diag{σ_1, ..., σ_{n_τ}}, σ_1 ≥ ... ≥ σ_{n_τ};
      V_{r,τ} := Q̃_τ id_{n_τ×k_τ} ∈ R^{τ×k_τ};
      for σ ∈ C_τ do Â_{τ,σ} := V_{r,τ}^H A|_{τ×σ}
    end
  else
    begin
      for τ' ∈ S(τ) do ComputeRowBasis(τ');
      Ĝ_τ := 0;
      for σ ∈ C_τ do
        begin
          Build Ā_{τ,σ} according to (5.13);
          Ĝ_τ := Ĝ_τ + Ā_{τ,σ} Ā_{τ,σ}^H
        end;
      Compute the Schur decomposition Q̃_τ^H Ĝ_τ Q̃_τ = D
        with D = diag{σ_1, ..., σ_{m_τ}}, σ_1 ≥ ... ≥ σ_{m_τ};
      Q_τ := Q̃_τ id_{m_τ×k_τ};
      for σ ∈ C_τ do Â_{τ,σ} := Q_τ^H Ā_{τ,σ};
      If necessary, copy B_{r,τ',τ} from Q_τ according to (5.9)
    end
end
```

5.3 Complexity

Now we aim to estimate the complexity of our algorithm. We recall the definition of C_L in condition 4 of Definition 2.1:

$$S(\tau) = \emptyset \Rightarrow n_\tau \leq C_L$$

for all leaves $\tau \in \mathcal{L}(\mathcal{T}(I))$.

We assume that there are sequences $(k_\ell)_{\ell\in\mathbb{N}}, (m_\ell)_{\ell\in\mathbb{N}}$ satisfying

$$k_\tau \leq k_{\mathrm{level}(\tau)} \quad \text{and} \quad m_\tau \leq m_{\mathrm{level}(\tau)}, \tag{5.14}$$

i.e., that k_τ and m_τ are bounded by constants depending only on the level of τ, and that there is a constant C_K satisfying

$$\begin{gathered}\sum_{\ell=0}^{\infty} 2^{\ell-\ell_{\max}} k_\ell \leq C_K, \quad \sum_{\ell=0}^{\infty} 2^{\ell-\ell_{\max}} m_\ell \leq C_K, \\ \sum_{\ell=0}^{\infty} 2^{\ell-\ell_{\max}} m_\ell^2 \leq C_K \quad \text{and} \quad \sum_{\ell=0}^{\infty} 2^{\ell-\ell_{\max}} k_\ell^3 \leq C_K\end{gathered} \tag{5.15}$$

Usually, k_ℓ and m_ℓ will be polynomials in ℓ and therefore fulfil this requirement.

We let $w_\ell := \#T_\ell(I)$ and assume that $\mathcal{T}(I)$ is an "almost balanced" binary tree, i.e., that there is a constant C_W satisfying

$$w_\ell \leq C_W 2^{\ell-\ell_{\max}} n.$$

5.3.1 Computational Complexity for $\widehat{G}_\tau$ and G_τ

Let us first consider the number of operations necessary for the computation of the Gram matrix G_τ for a leaf τ of $\mathcal{T}(I)$. The initialisation of G_τ requires $\mathcal{O}(n_\tau^2)$ operations, the computation of one of the increments $A|_{\tau\times\sigma} A|_{\tau\times\sigma}^H$ requires $\mathcal{O}(n_\tau^2 n_\sigma)$ operations for each $\sigma \in C_\tau$. Since C_τ is a partitioning of a subset of I, we have

$$\sum_{\sigma\in C_\tau} n_\sigma \leq n, \tag{5.16}$$

leading to a total complexity of $\mathcal{O}(n_\tau^2 n)$ for the computation of G_τ. Since τ is a leaf, we have $n_\tau \leq C_L$, so only a complexity of $\mathcal{O}(n)$ remains.

Let now $\tau \in T(I)$ be a node that is not a leaf of $\mathcal{T}(I)$. We estimate the number of operations necessary for the computation of $\widehat{G}_\tau$ from (5.12). The initialisation of $\widehat{G}_\tau$ requires $\mathcal{O}(m_\tau^2)$ operations, the computation of one of the increments $\widehat{A}_{\tau,\sigma}\widehat{A}_{\tau,\sigma}^H$ takes $\mathcal{O}(m_\tau^2 n_\sigma)$ operations. Due to (5.16), we end up with a complexity of $\mathcal{O}(m_\tau^2 n)$ for the computation of $\widehat{G}_\tau$.

If we denote the number of operations for the computation of G_τ (for leafs) or $\widehat{G}_\tau$ (for other clusters) by $E_{G,\tau}$, we get the estimate

$$E_{G,\tau} \leq C_G m_\tau^2 n$$

for a constant C_G. Summing up, we get the following estimate for the number of operations necessary for the computation for all $\tau \in T(I)$:

$$E_G := \sum_{\tau\in T(I)} E_{G,\tau} \leq C_G n \sum_{\tau\in T(I)} m_\tau^2 \leq C_G n \sum_{\ell=0}^{\ell_{\max}} \sum_{\tau\in T_\ell(I)} m_\tau^2$$

$$\leq C_G n \sum_{\ell=0}^{\ell_{\max}} w_l m_l^2 \leq C_G C_W n^2 \sum_{\ell=0}^{\ell_{\max}} 2^{\ell-\ell_{\max}} m_l^2 \leq C_G C_W C_K n^2. \tag{5.17}$$

5.3.2 Computational Complexity for $\tilde{Q}_\tau$

Now, we will derive an estimate for the number of operations necessary for the computation of $\tilde{Q}_\tau$, i.e., for solving the symmetric positive definite eigenproblem. We assume that an algorithm with cubic complexity is used, so the computation for a leaf $\tau \in T(I)$ takes $\mathcal{O}(n_\tau^3)$ operations, while the computation for the other nodes $\tau \in T(I)$ requires $\mathcal{O}(m_\tau^3)$ operations.

If we denote the number of operations for the computation of $\tilde{Q}_\tau$ once more by $E_{Q,\tau}$, we get the estimate

$$E_{Q,\tau} \leq C_Q m_\tau^3$$

for a constant C_Q. We sum up and get the estimate

$$\begin{aligned} E_Q &:= \sum_{\tau \in T(I)} E_{Q,\tau} \leq C_Q \sum_{\tau \in T(I)} m_\tau^3 \leq C_Q \sum_{\ell=0}^{\ell_{\max}} \sum_{\tau \in T_\ell(I)} m_\tau^3 \\ &\leq C_Q \sum_{\ell=0}^{\ell_{\max}} w_\ell m_\ell^3 \leq C_Q C_W n \sum_{\ell=0}^{\ell_{\max}} 2^{\ell-\ell_{\max}} m_\ell^3 \leq C_Q C_W C_K n. \end{aligned} \tag{5.18}$$

5.3.3 Computational Complexity for $\widehat{A}_{\tau,\sigma}$

To complete the complexity estimate, we consider the number of operations involved in the computation of the transformed matrices $\widehat{A}_{\tau,\sigma}$. The computation for a leaf $\tau \in T(I)$ and a $\sigma \in P_\tau^+$ (cf. (5.3)) takes $\mathcal{O}(k_\tau n_\tau n_\sigma)$ operations. By using (5.16), we get a complexity of $\mathcal{O}(k_\tau n_\tau n)$ for all updates corresponding to τ.

If $\tau \in T(I)$ is not a leaf of $\mathcal{T}(I)$, the computation requires $\mathcal{O}(k_\tau m_\tau n_\sigma)$ operations. By (5.16), the complexity for all updates corresponding to τ is of order $\mathcal{O}(k_\tau m_\tau n)$.

We denote by $E_{A,\tau}$ the number of operations for the computation of $\widehat{A}_{\tau,\sigma}$ for all $\sigma \in C_\tau$ by C_τ and get the estimate

$$E_{A,\tau} \leq C_{\mathrm{sp}} k_\tau m_\tau n$$

with a constant C_{sp}. For all $\tau \in T(I)$, the complexity is therefore bounded by

$$\begin{aligned} E_A &:= \sum_{\tau \in T(I)} E_{A,\tau} \leq C_{\mathrm{sp}} n \sum_{\tau \in T(I)} k_\tau m_\tau \leq C_{\mathrm{sp}} n \sum_{\ell=0}^{\ell_{\max}} \sum_{\tau \in T_\ell(I)} k_\tau m_\tau \\ &\leq C_{\mathrm{sp}} n \sum_{\ell=0}^{\ell_{\max}} w_\ell k_\ell m_\ell \leq C_{\mathrm{sp}} C_W n^2 \sum_{\ell=0}^{\ell_{\max}} 2^{\ell-\ell_{\max}} k_\ell m_\ell \leq C_{\mathrm{sp}} C_W C_K n^2. \end{aligned} \tag{5.19}$$

Summing up E_G, E_Q, and E_A, we find that the complexity of the algorithm is $\mathcal{O}(n^2)$.

5.4 $\mathcal{H}^2$-Approximation Error Estimate

In Subsection 5.2, we introduced an algorithm that computes consistent nested bases $\mathcal{V}_r$. The algorithm can be applied to A^H in order to get the bases $\mathcal{V}_c$.

This leaves us with the question if using both bases we can control the error of the $\mathcal{H}^2$-approximation of the matrix A.

We define Frobenius-orthogonal projections

$$\begin{aligned} \Pi_{\mathcal{V}_r} &: \mathbb{K}^{I\times I} \to \mathbb{K}^{I\times I}, \\ A = (A_b)_{b\in P_2} &\mapsto (\Pi_{\mathcal{V}_{r,\tau}} A_b)_{b=\tau\times\sigma\in P_2}, \end{aligned} \tag{5.20}$$

and

$$\begin{aligned} \Pi_{\mathcal{V}_c} &: \mathbb{K}^{I\times I} \to \mathbb{K}^{I\times I}, \\ A = (A_b)_{b\in P_2} &\mapsto (A_b \Pi_{\mathcal{V}_{c,\sigma}})_{b=\tau\times\sigma\in P_2}. \end{aligned} \tag{5.21}$$

They commute, and the product

$$\Pi := \Pi_{\mathcal{V}_r}\Pi_{\mathcal{V}_c} = \Pi_{\mathcal{V}_c}\Pi_{\mathcal{V}_r} \tag{5.22}$$

is the Frobenius-orthogonal projection onto $\mathcal{M}_{\mathcal{H}^2}(I\times I, P_2, \mathcal{V}_r, \mathcal{V}_c)$.

Due to the orthogonality of the projections, we get the following approximation error estimate:

Lemma 5.2 *Let $A \in \mathbb{K}^{I\times I}$. Then the inequality*

$$\|A - \Pi A\|_F^2 \leq \|A - \Pi_{\mathcal{V}_r} A\|_F^2 + \|A - \Pi_{\mathcal{V}_c} A\|_F^2$$

holds.

Proof. For a matrix $A \in \mathbb{K}^{I\times I}$, we have

$$\begin{aligned} \|A - \Pi A\|_F^2 &= \|A - \Pi_{\mathcal{V}_r}\Pi_{\mathcal{V}_c} A\|_F^2 = \|A - \Pi_{\mathcal{V}_r} A + \Pi_{\mathcal{V}_r} A - \Pi_{\mathcal{V}_r}\Pi_{\mathcal{V}_c} A\|_F^2 \\ &= \|(I - \Pi_{\mathcal{V}_r})A + \Pi_{\mathcal{V}_r}(A - \Pi_{\mathcal{V}_c} A)\|_F^2 \\ &= \|(I - \Pi_{\mathcal{V}_r})A\|_F^2 + \|\Pi_{\mathcal{V}_r}(A - \Pi_{\mathcal{V}_c} A)\|_F^2 \\ &\leq \|A - \Pi_{\mathcal{V}_r} A\|_F^2 + \|\Pi_{\mathcal{V}_r}\|_F^2 \|A - \Pi_{\mathcal{V}_c} A\|_F^2 \\ &\leq \|A - \Pi_{\mathcal{V}_r} A\|_F^2 + \|A - \Pi_{\mathcal{V}_c} A\|_F^2. \end{aligned}$$

■

Of course, we are interested in an optimal approximation. The following lemma states that quasi-optimal row and column bases lead to a quasi-optimal $\mathcal{H}^2$-representation:

Lemma 5.3 *Let $A \in \mathbb{K}^{I\times I}$. Then the inequality*

$$\max\{\|A - \Pi_{\mathcal{V}_r} A\|_F^2, \|A - \Pi_{\mathcal{V}_c} A\|_F^2\} \leq \|A - \Pi A\|_F^2$$

holds.

Proof. Due to (5.22), we have

$$\begin{aligned}\|A - \Pi_{\mathcal{V}_r} A\|_F^2 &= \|A\|_F^2 - \|\Pi_{\mathcal{V}_r} A\|_F^2 \leq \|A\|_F^2 - \|\Pi_{\mathcal{V}_c}\|_F^2 \|\Pi_{\mathcal{V}_r} A\|_F^2 \\ &\leq \|A\|_F^2 - \|\Pi_{\mathcal{V}_c} \Pi_{\mathcal{V}_r} A\|_F^2 = \|A\|_F^2 - \|\Pi A\|_F^2 = \|A - \Pi A\|_F^2.\end{aligned}$$

Swapping the roles of $\Pi_{\mathcal{V}_c}$ and $\Pi_{\mathcal{V}_r}$ completes the proof. ∎

6 Approximation of $\mathcal{H}$-Matrices

The complexity of the algorithm for the computation of nested row bases is optimal for full matrices: Since we have to consider n^2 matrix entries, any algorithm will be $\mathcal{O}(n^2)$.

Now we consider the computation of nested row bases for $\mathcal{H}$-matrices. We assume that the rank is "constant" (cf. (3.6)).

We require the partition P_2 to satisfy the condition

$$\#P_\tau \leq C_{\mathrm{sp}} \tag{6.1}$$

for a constant C_{sp} and all $\tau \in T(I)$ (cf. (5.2)).

Since a cluster $\tau \in T(I)$ can have no more than $\ell_{\max}$ father clusters, we conclude that

$$\#P_\tau^+ \leq C_{\mathrm{sp}} \ell_{\max}$$

(cf. (5.4)).

6.1 Efficient computation of $\widehat{G}_\tau$

We fix a matrix $A \in \mathcal{M}_{\mathcal{H}}(I \times I, P_2, k)$. Since $A|_b$ is a rank-k-matrix for each block $b = \tau \times \sigma \in P_2$, there are matrices $X_b \in \mathbb{K}^{\tau \times k_{\mathcal{H}}}$ and $Y_b \in \mathbb{K}^{\sigma \times k_{\mathcal{H}}}$ satisfying

$$A|_b = X_b Y_b^H. \tag{6.2}$$

We will make use of this structure in order to speed up the computation of the Gram matrices G_τ and $\widehat{G}_\tau$: We have

$$\begin{aligned}G_\tau &= \sum_{\sigma \in C_\tau} A|_{\tau \times \sigma} A|_{\tau \times \sigma}^H = \sum_{b \in P_\tau^+} (A|_b)|_{\tau \times \sigma} (A|_b)|_{\tau \times \sigma}^H \\ &= \sum_{\sigma \in C_\tau} X_b|_{\tau \times k_{\mathcal{H}}} Y_b^H Y_b X_b|_{\tau \times k_{\mathcal{H}}}^H,\end{aligned}$$

so we can compute $Z_b := Y_b^H Y_b$ for all blocks P_2 in advance. Since $\#T(I) = \mathcal{O}(n)$, we can conclude from (6.1) that $\#P_2 = \mathcal{O}(n)$, so the computation of all the $(Z_b)_{b \in P_2}$ can be accomplished in $\mathcal{O}(n k_{\mathcal{H}}^2)$ operations.

By using

$$\widehat{X}_{\tau,b} := V_{r,\tau}^H X_b|_{\tau \times k_{\mathcal{H}}}$$

instead of $\widehat{A}_{\tau,\sigma}$ (cf. (5.10)), we can rewrite (5.11) as

$$\begin{aligned}\widehat{G}_\tau = \widehat{A}_\tau \widehat{A}_\tau^H &= \begin{pmatrix} A_{\tau,\tau_1} A_{\tau,\tau_1}^H & \dots & A_{\tau,\tau_1} A_{\tau,\tau_s}^H \\ \vdots & \ddots & \vdots \\ A_{\tau,\tau_s} A_{\tau,\tau_1}^H & \dots & A_{\tau,\tau_s} A_{\tau,\tau_s}^H \end{pmatrix} \\ &= \sum_{b \in P_2^+} \begin{pmatrix} \widehat{X}_{\tau_1,b} Z_b \widehat{X}_{\tau_1,b}^H & \dots & \widehat{X}_{\tau_1,b} Z_b \widehat{X}_{\tau_s,b}^H \\ \vdots & \ddots & \vdots \\ \widehat{X}_{\tau_s,b} Z_b \widehat{X}_{\tau_1,b}^H & \dots & \widehat{X}_{\tau_s,b} Z_b \widehat{X}_{\tau_s,b}^H \end{pmatrix} \end{aligned} \tag{6.3}$$

for $S(\tau) = \{\tau_1, \dots, \tau_s\}$.

As in (5.13), we combine the matrices $\widehat{X}_{\tau',b}$ to get

$$\bar{X}_{\tau,b} := \begin{pmatrix} \widehat{X}_{\tau_1,b} \\ \vdots \\ \widehat{X}_{\tau_s,b} \end{pmatrix}, \tag{6.4}$$

so we can rewrite (6.3) in the form

$$\widehat{G}_\tau = \sum_{b \in P_2^+} \bar{X}_{\tau,b} \begin{pmatrix} Z_b & & \\ & \ddots & \\ & & Z_b \end{pmatrix} \bar{X}_{\tau,b}^H. \tag{6.5}$$

The computation of $\widehat{G}_\tau$ therefore requires only $\mathcal{O}(m_\tau k_{\mathcal{H}} (m_\tau + k_{\mathcal{H}}) \ell_{\max})$ operations, while in the case of the general matrix $\mathcal{O}(m_\tau^2 n)$ operations were necessary.

Due to the equation $\widehat{X}_{\tau,b} = Q_\tau^H \bar{X}_{\tau,b}$, we can use $\bar{X}_{\tau,b}$ to compute $\widehat{X}_{\tau,b}$ efficiently, i.e., in only $\mathcal{O}(k_\tau m_\tau k_{\mathcal{H}})$ operations, while in the case of a general matrix $\mathcal{O}(k_\tau m_\tau n)$ operations are necessary.

6.2 Algorithm

The computation of the auxiliary matrices Z_b is straightforward, so we only give the recursive algorithm for the computation of a nested row bases for a given $\mathcal{H}$-matrix:

```
procedure ComputeRowBasisHMatrix(τ);
begin
  if S(τ) = ∅ then
    begin
      G_τ := 0;
      for b ∈ P_τ^+ do G_τ := G_τ + X_b|_{τ×k_H} Z_b X_b|_{τ×k_H}^H;
      Compute the Schur decomposition Q̃_τ^H G_τ Q̃_τ = D
        with D = diag{σ_1, ..., σ_{n_τ}}, σ_1 ≥ ... ≥ σ_{n_τ};
      V_{r,τ} := Q̃_τ id_{n_τ × k_τ} ∈ R^{τ×k_τ};
      for b ∈ P_τ^+ do X̂_{τ,b} := V_{r,τ}^H X_b|_{τ×k_H}
```

```
    end
  else
    begin
      for τ' ∈ S(τ) do ComputeRowBasis(τ');
      Ĝ_τ := 0;
      for b ∈ P_τ^+ do
        begin
          Build X̃_{τ,b} according to (6.4);
          Compute Ĝ_τ according to (6.5)
        end;
      Compute the Schur decomposition Q̃_τ^H Ĝ_τ Q̃_τ = D
        with D = diag{σ_1, ..., σ_{m_τ}}, σ_1 ≥ ... ≥ σ_{m_τ};
      Q_τ := Q̃_τ id_{m_τ × k_τ};
      for b ∈ P_τ^+ do X̂_{τ,b} := Q_τ^H X̃_{τ,b};
      If necessary, copy B_{r,τ',τ} from Q_τ according to (5.9)
    end
end
```

6.3 Complexity

The computation of $\tilde{Q}_\tau$ in the $\mathcal{H}$-matrix algorithm requires the same number of operations as in the full matrix algorithm, i.e., $\mathcal{O}(n)$ operations.

The computation of $\widehat{X}_{\tau,b}$ takes $\mathcal{O}(k_\tau m_\tau k_{\mathcal{H}})$ operations in the $\mathcal{H}$-matrix case and $\mathcal{O}(k_\tau m_\tau n)$ operations in the full matrix case. Proceeding as in Subsection 5.3.3, we find that the computation for all clusters $\tau \in T(I)$ and all blocks $b \in P_2$ requires $\mathcal{O}(nk_{\mathcal{H}})$ operations.

As stated above, the computation of $\widehat{G}_\tau$ requires $\mathcal{O}(m_\tau k_{\mathcal{H}}(m_\tau + k_{\mathcal{H}})\ell_{\max})$ operations, i.e., there is a constant C'_G satisfying

$$E'_{G,\tau} \leq C'_G m_\tau k_{\mathcal{H}}(m_\tau + k_{\mathcal{H}})\ell_{\max},$$

where $E'_{G,\tau}$ denotes the number of operations involved in the computation of $\widehat{G}_\tau$. Summing up, we get

$$\begin{aligned}
E'_G := \sum_{\tau\in T(I)} E'_{G,\tau} &\leq C'_G k_{\mathcal{H}}\ell_{\max} \sum_{\tau\in T(I)} m_\tau^2 + C'_G k_{\mathcal{H}}^2 \ell_{\max} \sum_{\tau\in T(I)} m_\tau \\
&\leq C'_G k_{\mathcal{H}}\ell_{\max} \sum_{\ell=0}^{\ell_{\max}} \sum_{\tau\in T_\ell(I)} m_\ell^2 + C'_G k_{\mathcal{H}}^2 \ell_{\max} \sum_{\ell=0}^{\ell_{\max}} \sum_{\tau\in T_\ell(I)} m_\ell \\
&\leq C'_G k_{\mathcal{H}}\ell_{\max} \sum_{\ell=0}^{\ell_{\max}} w_\ell m_\ell^2 + C'_G k_{\mathcal{H}}^2 \ell_{\max} \sum_{\ell=0}^{\ell_{\max}} w_\ell m_\ell^2 \\
&\leq C'_G C_W k_{\mathcal{H}}\ell_{\max} n \sum_{\ell=0}^{\ell_{\max}} 2^{\ell-\ell_{\max}} m_\ell^2 + C'_G C_W k_{\mathcal{H}}^2 \ell_{\max} n \sum_{\ell=0}^{\ell_{\max}} 2^{\ell-\ell_{\max}} m_\ell
\end{aligned}$$

$$\leq C'_G C_W C_K k_{\mathcal{H}}(1+k_{\mathcal{H}})\ell_{\max} n,$$

so the complexity of the entire algorithm is in $\mathcal{O}(n\ell_{\max}k_{\mathcal{H}}^2)$.

7 Adaptivity

As mentioned in Remark 5.1, it is possible to choose the rank k_τ in order to reach a given precision ϵ_τ, i.e., to satisfy the condition

$$\|A_\tau - \Pi_{\mathcal{V}_{r,\tau}} A_\tau\|_F^2 = \sum_{\nu=k_\tau+1}^{m_\tau} \sigma_\nu^2 =: e_\tau^2 \leq \epsilon_\tau^2. \tag{7.1}$$

The following Lemma combines the cluster-oriented quantities e_τ to form the global error:

Lemma 7.1 *Let P_2 be a partitioning and let $\mathcal{V}_r = (\mathcal{V}_{r,\tau})_{\tau\in T(I)}$ be a consistent family of spaces. Then the equation*

$$\|A - \Pi_{\mathcal{V}_r} A\|_F^2 = \sum_{\tau\in T(I)} e_\tau^2. \tag{7.2}$$

holds for the Frobenius-orthogonal projection $\Pi_{\mathcal{V}_r}$ introduced in (5.20).

Proof. We consider auxiliary partitionings

$$P_2^\ell := \left\{ b \in T(I\times I) \ : \ (\mathrm{level}(b) > \ell \wedge b \in P_2) \vee \left(\mathrm{level}(b) = \ell \wedge \exists b' \in P_2 \ : \ b \subseteq b'\right)\right\},$$

with auxiliary Frobenius-orthogonal projections

$$\begin{aligned}\Pi_{\mathcal{V}_r,\ell} &: \mathbb{K}^{I\times I} \to \mathbb{K}^{I\times I},\\ A = (A_b)_{b\in P_2^\ell} &\mapsto (\Pi_{V_{r,\tau}} A_b)_{b=\tau\times\sigma\in P_2^\ell}.\end{aligned}$$

The partitionings for a standard example can be found in Figure 3.

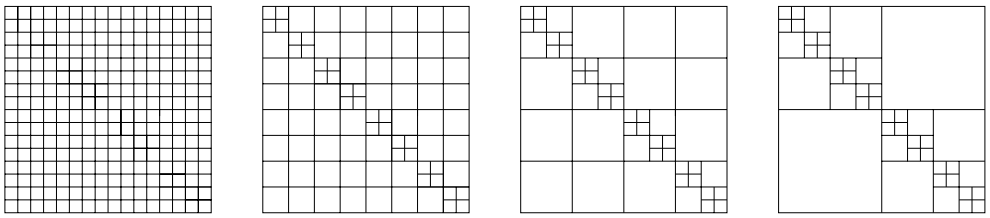

Figure 3: Partitionings P_2^3, P_2^2, P_2^1 and $P_2^0 = P_2$ for the example from Figure 2

We have $P_2^0 = P_2$ and $P_2^{\ell_{\max}} = T_{\ell_{\max}}(I\times I)$ and, consequently, $\Pi_{\mathcal{V}_r,0} = \Pi_{\mathcal{V}_r}$ and $\Pi_{\mathcal{V}_r,\ell_{\max}} = I$. Since the family $\mathcal{V}_r$ is consistent, the equation $\Pi_{\mathcal{V}_r,\ell_1}\Pi_{\mathcal{V}_r,\ell_2} = \Pi_{\mathcal{V}_r,\min\{\ell_1,\ell_2\}}$ holds for all $\ell_1,\ell_2 \in \{0,\ldots,\ell_{\max}\}$, therefore we get

$$\|A - \Pi_{\mathcal{V}_r,\ell}\|_F^2 = \|A - \Pi_{\mathcal{V}_r,\ell+1}A + \Pi_{\mathcal{V}_r,\ell+1}A - \Pi_{\mathcal{V}_r,\ell}A\|_F^2$$

$$= \|(I - \Pi_{\mathcal{V}_r,\ell+1})A + \Pi_{\mathcal{V}_r,\ell+1}(I - \Pi_{\mathcal{V}_r,\ell})A\|_F^2$$
$$= \|A - \Pi_{\mathcal{V}_r,\ell+1}\|_F^2 + \|\Pi_{\mathcal{V}_r,\ell+1}A - \Pi_{\mathcal{V}_r,\ell}A\|_F^2.$$

Since the error introduced by changing from the $\mathcal{H}^2$-matrix approximation corresponding to the partitioning $P_2^{\ell+1}$ to that corresponding to P_2^ℓ is given by the equation

$$\|\Pi_{\mathcal{V}_r,\ell+1}A - \Pi_{\mathcal{V}_r,\ell}A\|_F^2 = \sum_{\tau\in T_\ell(I)} e_\tau^2,$$

we get the desired equality (7.2) by summing up these terms for all $\ell \in \{0,\dots,\ell_{\max}\}$.
■

A straightforward approach to computing an $\mathcal{H}^2$-matrix for a given matrix $A \in \mathbb{K}^{I\times I}$ would be to set

$$\epsilon_\tau := \frac{\tilde{\epsilon}}{\sqrt{2\#T(I)}} \leq \frac{\tilde{\epsilon}}{\sqrt{4n}}$$

for a fixed $\tilde{\epsilon} \in \mathbb{R}_{>0}$ and choose k_τ in order to satisfy (7.1), leading to

$$\|A - \Pi_{\mathcal{V}_r} A\|_F^2 = \sum_{\tau\in T(I)} e_\tau^2 \leq \sum_{\tau\in T(I)} \epsilon_\tau^2 = \frac{\tilde{\epsilon}^2 \#T(I)}{2\#T(I)} = \tilde{\epsilon}^2/2.$$

Applying the algorithm for the computation of the column bases, we get

$$\|A - \Pi A\|_F^2 \leq \tilde{\epsilon}^2$$

by Lemma 5.2.

Obviously, feeding any other vector $\epsilon = (\epsilon_\tau)_{\tau\in T(I)}$ satisfying

$$\sqrt{\sum_{\tau\in T(I)} \epsilon_\tau^2} \leq \tilde{\epsilon}/\sqrt{2}$$

into our algorithm will yield the same approximation error. Therefore, we are looking for a vector $\epsilon \in \mathbb{R}_{\geq 0}^{T(I)}$ with $\|\epsilon\|_2 \leq \tilde{\epsilon}/\sqrt{2}$ that minimises a certain cost functional.

If, for example, matrix-vector multiplications are very important for the application in question, the functional

$$C(\epsilon) := \sum_{\tau\in T(I)} k_\tau(\epsilon)^2,$$

where $k_\tau(\epsilon)$ is the rank required to satisfy the condition (7.1), should be minimized.

The functions $k_\tau(\epsilon)$ depend not only on ϵ_τ, but also on the values of $\epsilon_{\tau'}$ for any descendant of τ in the tree $\mathcal{T}(I)$, so finding an optimal ϵ usually means solving a complicated nonlinear minimisation problem on the ball with radius $\tilde{\epsilon}/\sqrt{2}$.

8 Examples

In this last section, we will investigate the properties of the adaptive $\mathcal{H}^2$-approximation algorithm.

8.1 Approximation of an Integral Operator

Since the basic idea of the $\mathcal{H}^2$-matrix concept is the panel clustering technique for integral equations, our first example investigates the approximation of the single layer potential corresponding to Poisson's equation in 2D. The kernel function

$$k(x,y) := \log \|x-y\|$$

is discretised on a polygonal closed curve using a Galerkin projection with piecewise constant basis functions.

8.1.1 Constant Rank

We apply the greedy-type algorithm of Subsection 5.2 to the matrices corresponding to the Galerkin discretisation and, for the first test, choose a fixed rank of $k_{\text{const}} = 4$. In order to speed up the algorithm, the maximal size of leaves C_L is set to 8.

The relative approximation errors

$$\text{Error}_F := \frac{\|A - \Pi A\|_F}{\|A\|_F}, \quad \text{Error}_2 := \frac{\|A - \Pi A\|_2}{\|A\|_2}$$

in the Frobenius and Euclidean norm are reported in the Tables 1 and 2, along with the time[2] needed for the conversion from full to $\mathcal{H}^2$-matrix, the time required for one matrix-vector multiplication, the amount of memory needed to store all the data corresponding to the $\mathcal{H}^2$-matrix and the compression factor achieved in comparison to full storage.

8.1.2 Adaptive Rank

As outlined in Section 7, our algorithm cannot only select the $\mathcal{H}^2$-basis functions adaptively, but it can also be used to determine the ranks for each cluster in order to achieve a given precision.

If we require the approximation to be accurate up to a relative Frobenius error of $\epsilon = 10^{-6}$ and apply the algorithm to the BEM matrices corresponding to an approximation of the unit circle and to the boundary of the unit square, we get the rank distributions listed in the Tables 3 and 4.

In both cases, we observe a slow monotonous growth in the rank when passing from smaller to larger clusters. Since the large clusters on level $\ell = 2$ appear only once in the partitioning P_2 while all other clusters appear three times, a slightly reduced rank is

[2] All computations are performed on a SUN Enterprise server with an UltraSPARC 2 processor running at 248 MHz

n	256	512	1024	2048	4096	8192	16384	32768
Conv/s	0.09	0.27	1.15	4.44	17.22	70.90	343.0	1415
		(3.0)	(4.3)	(3.9)	(3.9)	(4.1)	(4.8)	(4.1)
MVM/ms	1.11	2.22	4.54	9.67	21.12	42.78	107.6	219.9
		(2.0)	(2.0)	(2.2)	(2.2)	(2.0)	(2.5)	(2.0)
MVM/MFlops	0.03	0.06	0.12	0.24	0.48	0.96	1.92	3.84
		(2.0)	(2.0)	(2.0)	(2.0)	(2.0)	(2.0)	(2.0)
Mem/KB	142.7	288.9	581.4	1166	2336	4676	9356	18716
		(2.0)	(2.0)	(2.0)	(2.0)	(2.0)	(2.0)	(2.0)
Comp	0.28	0.14	0.07	0.04	0.02	0.01	0.004	0.002
$\mathrm{Err}_F/10^{-5}$	2.86	3.43	3.66	3.74	3.77	3.79	3.79	3.79
$\mathrm{Err}_2/10^{-5}$	1.38	1.45	1.46	1.45	1.48	1.48	1.48	1.48

Table 1: $\mathcal{H}^2$-approximation of the Poisson single layer potential on the unit circle for constant rank $k_{\mathrm{const}} = 4$. The ratios of consecutive columns are given in brackets.

sufficient to approximate the corresponding blocks leading to the only exception to the otherwise monotonous growth.

Apparently, the adaptive procedure compensates the reduced "smoothness" of the matrix corresponding to the boundary of the unit square by slightly increasing the rank for the larger clusters.

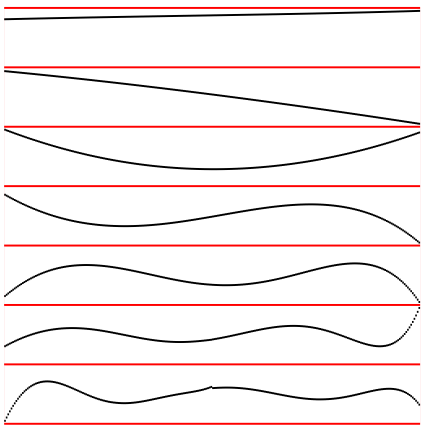

Figure 4: Basis chosen for a level 3 cluster

The seven basis functions chosen by the algorithm for a level 3 cluster can be seen in Figure 4: They resemble piecewise constant approximations of the first seven Legendre polynomials.

n	256	512	1024	2048	4096	8192	16384	32768
Conv/s	0.08	0.26	1.14	4.48	17.48	74.69	331.9	1391
		(3.3)	(4.4)	(3.9)	(3.9)	(4.3)	(4.4)	(4.2)
MVM/ms	1.06	2.23	4.53	9.59	21.05	43.24	107.1	216.4
		(2.1)	(2.0)	(2.1)	(2.2)	(2.1)	(2.5)	(2.0)
MVM/MFlops	0.03	0.06	0.12	0.24	0.48	0.96	1.92	3.84
		(2.0)	(2.0)	(2.0)	(2.0)	(2.0)	(2.0)	(2.0)
Mem/KB	142.7	288.9	581.4	1166	2336	4676	9356	18716
		(2.0)	(2.0)	(2.0)	(2.0)	(2.0)	(2.0)	(2.0)
Comp	0.28	0.14	0.07	0.04	0.02	0.01	0.004	0.002
$\mathrm{Err}_F/10^{-5}$	7.85	8.01	8.01	7.98	7.96	7.95	7.94	7.93
$\mathrm{Err}_2/10^{-5}$	5.10	5.12	5.10	5.08	5.07	5.06	5.06	5.06

Table 2: $\mathcal{H}^2$-approximation of the Poisson single layer potential on the boundary of the unit square for constant rank $k_{\mathrm{const}} = 4$. The ratios of consecutive columns are given in brackets.

8.2 Approximation of a finite elements inverse

For the next example, we consider the $\mathcal{H}^2$-approximation of the inverse of a finite element matrix corresponding to Laplace's operator discretised using piecewise linear functions on a regular triangulation of the unit square. The discretisation typically leads to matrices with dimensions that are much larger than those considered in the BEM case, so we cannot store the entire matrix in a fully populated format and apply the algorithm `ComputeRowBasis`.

Instead, we use the approximate inversion routines of Grasedyck [1] to compute an $\mathcal{H}$-matrix approximation $A_{\mathcal{H}}^{-1}$ of the matrix A^{-1} (with blockwise rank of $k_{\mathcal{H}} = 8$) which is then converted to an $\mathcal{H}^2$-matrix $A_{\mathcal{H}^2}^{-1}$ using the algorithm `ComputeRowBasisHMatrix`.

The rank for the $\mathcal{H}^2$-matrix is not chosen by our adaptive procedure, but according to equation (3.7) with $\alpha = \beta = 2$. In Table 5, the time for the $\mathcal{H}$-inversion, for the conversion to $\mathcal{H}^2$-format, the number of flops for $\mathcal{H}$- and $\mathcal{H}^2$-matrix-vector multiplications, the memory requirements for both matrix representations and the relative errors

$$\mathrm{err}_{\mathcal{H}} := \|I - A_{\mathcal{H}}^{-1} A\|_2 \quad \text{and} \quad \mathrm{err}_{\mathcal{H}^2} := \|I - A_{\mathcal{H}^2}^{-1} A\|_2$$

are reported. Obviously, the $\mathcal{H}^2$-matrices require less memory and allow faster matrix-vector-multiplications compared with $\mathcal{H}$-matrices.

n	256	512	1024	2048	4096	8192	16384	32768
2	5	5	5	5	5	5	6	6
3	6	6	6	6	6	7	7	7
4	6	6	6	6	6	7	7	7
5		6	6	6	6	6	7	7
6			6	6	6	6	6	6
7				6	6	6	6	6
8					5	6	6	6
9						5	5	6
10							5	5
11								5
Conv/s	0.11	0.38	1.54	5.91	22.75	103.3	419.9	1627
		(3.5)	(4.1)	(3.8)	(3.8)	(4.5)	(4.1)	(3.9)
MVM/ms	1.28	2.88	5.62	12.08	22.63	53.88	119.5	243.3
		(2.3)	(2.0)	(2.1)	(1.9)	(2.4)	(2.2)	(2.0)
MVM/MFlops	0.04	0.07	0.15	0.29	0.57	1.13	2.22	4.42
		(1.8)	(2.1)	(1.9)	(2.0)	(1.8)	(2.0)	(2.0)
Mem/KB	167.8	347.0	705.5	1391	2703	5393	10606	21167
		(2.1)	(2.0)	(2.0)	(1.9)	(2.0)	(2.0)	(2.0)
Comp	0.33	0.17	0.09	0.04	0.02	0.01	0.005	0.002
$\mathrm{Err}_F/10^{-6}$	0.09	0.19	0.04	0.32	0.42	0.57	0.66	1.21
$\mathrm{Err}_2/10^{-6}$	0.05	0.05	0.06	0.13	0.09	0.10	0.06	0.05

Table 3: Adaptively chosen rank distribution across cluster levels for the Poisson single layer potential on the unit circle. The ratios of consecutive columns are given in brackets.

n	256	512	1024	2048	4096	8192	16384	32768
2	6	7	7	7	7	7	7	7
3	7	7	7	8	8	8	8	8
4	6	7	7	7	7	7	7	7
5		6	6	7	7	7	7	7
6			6	6	6	7	7	7
7				6	6	6	6	7
8					6	6	6	6
9						6	6	6
10							5	6
11								5
Conv/s	0.11	0.38	1.59	5.94	23.04	104.5	424.1	1637
		(3.5)	(4.2)	(3.7)	(3.9)	(4.5)	(4.1)	(3.9)
MVM/ms	1.28	2.95	5.58	12.13	22.68	53.34	122.6	249.4
		(2.3)	(1.9)	(2.2)	(1.9)	(2.4)	(2.3)	(2.0)
MVM/MFlops	0.04	0.08	0.15	0.29	0.57	1.13	2.22	4.41
		(2.0)	(1.9)	(1.9)	(2.0)	(2.0)	(2.0)	(2.0)
Mem/KB	173.6	356.6	716.8	1392	2734	5388	10627	21109
		(2.1)	(2.0)	(1.9)	(2.0)	(2.0)	(2.0)	(2.0)
Comp	0.34	0.17	0.09	0.04	0.02	0.01	0.005	0.003
$\mathrm{Err}_F/10^{-6}$	0.34	0.30	0.27	0.47	0.29	0.44	0.73	0.62
$\mathrm{Err}_2/10^{-6}$	0.19	0.16	0.13	0.14	0.09	0.10	0.06	0.06

Table 4: Adaptively chosen rank distribution across cluster levels for the Poisson single layer potential on the boundary of the unit square. The ratios of consecutive columns are given in brackets.

	4096	10000	16384	40000	65536
Time for Inversion/secs	174	878.8	1360	5324	8426
Time for Conversion/secs	3.6	10.3	21.5	62.9	120
$\mathcal{H}$-MVM/MFlops	10.8	25.7	59.2	133.7	293.4
$\mathcal{H}$-Memory/MBytes	42.4	102.0	231.9	530.6	1151
$\mathcal{H}$-Error	0.000022	0.000054	0.000115	0.000382	0.000645
$\mathcal{H}^2$-MVM/MFlops	8.9	16.3	41.6	70.7	179.5
$\mathcal{H}^2$-Memory/MBytes	33.9	62.9	159.4	274.0	689
$\mathcal{H}^2$-Error	0.000076	0.000602	0.000139	0.000649	0.000646

Table 5: $\mathcal{H}^2$-approximation of the inverse of the discretised Laplace operator on the unit square

References

[1] L. Grasedyck: *Theorie und Anwendungen Hierarchischer Matrizen.* PhD thesis, University of Kiel, Germany, 2001.

[2] W. Hackbusch: *Iterative Solution of Large Sparse Systems.* Springer Verlag, New York, 1994.

[3] W. Hackbusch: *Integral Equations. Theory and Numerical Treatment.* ISNM 128. Birkhäuser, Basel, 1995.

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

[5] W. Hackbusch and B. N. Khoromskij: *A sparse $\mathcal{H}$-matrix arithmetic. Part II: application to multi-dimensional problems.* Computing **64** (2000) 21–47

[6] W. Hackbusch, B. N. Khoromskij, and S. Sauter: *On $\mathcal{H}^2$-matrices.* Lectures on Applied Mathematics (Bungartz, Hoppe, Zenger, eds.), Springer Verlag, Heidelberg, 2000

[7] W. Hackbusch and Z. P. Nowak: *On the fast matrix multiplication in the boundary element method by panel clustering.* Numer. Math. **54** (1989) 463–491.

[8] S. A. Sauter: *Über die effiziente Verwendung des Galerkin-Verfahrens zur Lösung Fredholmscher Integralgleichungen.* Dissertation. Universität Kiel. 1992.

[9] S. A. Sauter: *Variable order panel clustering.* Computing **64** (2000) 223–261