Chapter 2

Hierarchical Matrices by Interpolation

2.1 Cluster Tree (12.03.2009)

Let $\mathcal{I} = \{1, \ldots, n\}$ be the non-empty index set. The elements of $\mathcal{I}$ are called **degrees of freedom**. The non-empty subsets $\sigma \subseteq \mathcal{I}$ are referred to as **clusters**. Moreover, we have to provide the following geometrical information in $\mathbb{R}^d$ for the degrees of freedom:

- For each index $j \in \mathcal{I}$, let $\operatorname{supp}(j) \subseteq \mathbb{R}^d$ be an associated set in $\mathbb{R}^d$.
- For each cluster $\sigma \subseteq \mathcal{I}$, we define $\cup\sigma := \bigcup_{j\in\sigma} \operatorname{supp}(j)$.
- Moreover, Algorithm 2.2 needs that we fix an element $x_j \in \operatorname{supp}(j)$ for each degree of freedom $j \in \mathcal{I}$, e.g., the center of mass of $\operatorname{supp}(j)$.

Moreover, for easier numerical handling we will replace $\cup\sigma$ in praxis by an axis-oriented box:

- For each cluster σ we fix the **bounding box** $B_\sigma = \prod_{j=1}^{d} [a_j, b_j]$ defined as the axis oriented box of minimal size containing $\cup\sigma$, i.e. $\cup\sigma \subseteq B_\sigma$.

The following examples will hopefully illustrate what we are thinking of.

Example. (i) Let $\mathcal{T} = \{T_1, \ldots, T_n\}$ be a partition of the domain $\Omega \subseteq \mathbb{R}^d$ and consider the space $\mathcal{P}^0(\mathcal{T})$ of all $\mathcal{T}$-piecewise constant functions. The basis functions of $\mathcal{P}^0(\mathcal{T})$ are just the characteristic functions χ_{T_j} for $j = 1, \ldots, n$. For $j \in \mathcal{I}$, the corresponding set is just $\operatorname{supp}(j) := \operatorname{supp}(\chi_{T_j}) = \overline{T_j}$, whence $\cup\sigma = \bigcup_{j\in\sigma} \overline{T_j}$ for any cluster $\sigma \subseteq \mathcal{I}$. For $T_j \in \mathcal{T}$, let $x_j \in T_j$ denote, e.g., the center of the element T_j.
(ii) Let $\mathcal{T}$ be a regular triangulation of a domain $\Omega \subseteq \mathbb{R}^d$. We consider the space $\mathcal{S}^1(\mathcal{T})$ consisting of all $\mathcal{T}$-piecewise affine and globally continuous functions. A convenient basis of $\mathcal{S}^1(\mathcal{T})$ is given by the hat functions ϕ_j. If $\mathcal{N} := \{z_1, \ldots, z_n\}$ denotes the set of all nodes of $\mathcal{T}$, $\phi_j \in \mathcal{S}^1(\mathcal{T})$ is uniquely defined by the Kronecker property $\phi_j(z_k) = \delta_{jk}$. In this case, we define $x_j := z_j$ and $\operatorname{supp}(j) := \operatorname{supp}(\phi_j)$ for all $j \in \mathcal{I}$. □

For the ease of presentation, we define cluster trees as binary trees, i.e. each node in the tree that is not a leaf, has precisely two sons. This assumption can be weakened easily. For notational simplicity, we identify the tree with the set of its nodes.

Definition. $\mathbb{T}_\mathcal{I}$ is a **cluster tree** for the index set $\mathcal{I}$ and the parameter $C_{\text{leaf}} \in \mathbb{N}$, provided
(i) $\mathcal{I}$ is the root of $\mathbb{T}_\mathcal{I}$,
(ii) each node $\sigma \in \mathbb{T}_\mathcal{I}$ is a non-empty subset of $\mathcal{I}$,
(iii) if $\sigma \in \mathbb{T}_\mathcal{I}$ is a leaf, then $|\sigma| \leq C_{\text{leaf}}$,
(iv) if $\sigma \in \mathbb{T}_\mathcal{I}$ is not a leaf, then there are two *unique* (non-empty) clusters $\sigma', \sigma'' \in \mathbb{T}_\mathcal{I}$ with $\sigma = \sigma' \cup \sigma''$ and $\sigma' \cap \sigma'' = \emptyset$, called **sons**. We write $\text{sons}(\sigma) = \{\sigma', \sigma''\}$.

The constant C_{leaf} is a pay-off constant: The smallest admissible blocks in an $\mathcal{H}$-matrix will be of size $C_{\text{leaf}} \times C_{\text{leaf}}$. If the block is too small, the approximate computation of it is more costly than the exact assembling. For my computations, I choose C_{leaf} between 20 and 100.

The following lemma proves that the number of nodes in a cluster tree depends only linearly on the number $n = |\mathcal{I}|$ of indices. For notational convenience, we denote by $\text{leaves}(\mathbb{T}_\mathcal{I})$ the set of all leaves of the cluster tree $\mathbb{T}_\mathcal{I}$.

Lemma 2.1. *For a cluster tree $\mathbb{T}_\mathcal{I}$, there holds*

$$|\mathbb{T}_\mathcal{I}| \leq 2\,|\text{leaves}(\mathbb{T}_\mathcal{I})| - 1 \leq 2n - 1, \tag{2.1}$$

i.e. the number of nodes in the tree is bounded by the number of indices.

Proof. The lemma is proved by induction on the number $n = |\mathcal{I}|$ of indices. The claim is obviously true for $n = 1$. For $n > 1$ and $|\text{leaves}(\mathbb{T}_\mathcal{I})| > 1$, we consider the sons σ, τ of the root $\mathcal{I}$ and apply the induction hypothesis for the branches with σ, τ as roots to obtain

$$|\mathbb{T}_\mathcal{I}| = |\mathbb{T}_\sigma| + |\mathbb{T}_\tau| + 1 \leq \big(2\,|\text{leaves}(\mathbb{T}_\sigma)| - 1\big) + \big(2\,|\text{leaves}(\mathbb{T}_\tau)| - 1\big) + 1 = 2\,|\text{leaves}(\mathbb{T}_\mathcal{I})| - 1,$$

where we have used that the branches of a cluster tree are disjoint. ■

The following recursive function is initially called with

`CreateClusterTree(`$\emptyset$`,`$\mathcal{I}$`)`.

One obtains a cluster tree simply by splitting the bounding boxes.

Algorithm 2.2 (Create Cluster Tree).

```
function CreateClusterTree(var T_I,σ)
if |σ| ≤ C_leaf
   return
else
   Create axis oriented box B_σ of minimal size with ∪σ ⊆ B_σ
   Split B_σ along longest edge into boxes Q', Q''
   Define sons σ', σ'' of σ by
      σ' = {I ∈ σ | x_I ∈ Q'},   σ'' = σ\σ'.
   Add σ' to T_I and call CreateClusterTree(T_I,σ')
   Add σ'' to T_I and call CreateClusterTree(T_I,σ'')
end
```

Remark. (i) For pathological triangulations it may happen that the algorithm fails because $\sigma' \in \{\sigma, \emptyset\}$. Thus, an implementation needs some minor modifications.
(ii) We stress that the boxes Q' and Q'' are, in general, not the bounding boxes of σ' and σ'', respectively. □

Remark. To minimize the storage requirements for the clustertree, one usually sorts the index set and then stores only the first index of a cluster $\sigma \in \mathbb{T}_\mathcal{I}$ as well as its length $|\sigma|$. □

Definition. For a cluster tree $\mathbb{T}_\mathcal{I}$, we inductively define the **level function**

$$\text{level} : \mathbb{T}_\mathcal{I} \to \mathbb{N}_0$$

by $\text{level}(\mathcal{I}) = 0$ and $\text{level}(\sigma') := \text{level}(\sigma) + 1$ for $\sigma' \in \text{sons}(\sigma)$. The **depth of a cluster tree** is

$$\text{depth}(\mathbb{T}_\mathcal{I}) := \max_{\sigma \in \mathbb{T}_\mathcal{I}} \text{level}(\sigma).$$

For uniform meshes one can prove that Algorithm 2.2 creates a cluster tree with $\text{depth}(\mathbb{T}_\mathcal{I}) = \mathcal{O}(\log(n))$ due to the binary splitting of the index set. However, for pathological examples like the following one can in fact observe $\text{depth}(\mathbb{T}_\mathcal{I}) = \mathcal{O}(n)$:

Example. We consider the interval $[0, 1)$ with a partition into $n+1$ subintervals $T_k := [x_{k-1}, x_k)$ for $k = 1, \ldots, n+1$, where the nodes x_j are given inductively: $x_0 := 0$, $x_j := x_{j-1} + (1/2)^j$ for $j = 1, \ldots, n$, and $x_{n+1} := 1$. Then, the cluster tree generated by Algorithm 2.2 satisfies $\text{depth}(\mathbb{T}_\mathcal{I}) = n$. □

2.2 Block Partitioning (19.05.2009)

The starting point for this section is the following:

- Let $\mathbb{T}_\mathcal{I}$ be a cluster tree for the index set $\mathcal{I} = \{1, \ldots, m\}$.
- Let $\mathbb{T}_\mathcal{J}$ be a cluster tree for the index set $\mathcal{J} = \{1, \ldots, n\}$.

The next step is to derive a hierarchical partitioning of $\mathcal{I} \times \mathcal{J}$. Note the index set $\mathcal{I} \times \mathcal{J}$ corresponds to an $m \times n$-matrix, i.e. we are constructing some (hierarchical) block structure for some matrix $A \in \mathbb{K}^{m \times n}$.

According to the introduction, we need a criterion on which blocks $A|_{\sigma \times \tau}$ one may allow a low-rank approximation. This is done via a so-called **admissibility condition**. In the following, we present an admissibility condition (2.2) which is used for the interpolation approach for kernels $\kappa(x, y)$ which are smooth for $x \neq y$. This condition allows the uniform control of the interpolation error which will be proven in Section 2.4–2.6 below.

Definition. Let $\sigma \in \mathbb{T}_{\mathcal{I}}$ and $\tau \in \mathbb{T}_{\mathcal{J}}$ be clusters with bounding boxes $B_\sigma, B_\tau \subseteq \mathbb{R}^d$. The pair (σ, τ) is called **block**. For a fixed parameter $\eta > 0$, the block (σ, τ) is **admissible** provided

$$\min\{\operatorname{diam}(B_\sigma), \operatorname{diam}(B_\tau)\} \leq \eta \operatorname{dist}(B_\sigma, B_\tau). \tag{2.2}$$

For a partitioning $\mathbb{P} \subseteq \mathbb{T}_{\mathcal{I}} \times \mathbb{T}_{\mathcal{J}}$ of $\mathcal{I} \times \mathcal{J}$, we define the **far field**

$$\mathbb{P}_{\text{far}} := \big\{(\sigma, \tau) \in \mathbb{P} \,\big|\, (\sigma, \tau) \text{ is admissible}\big\} \tag{2.3}$$

and the **near field**

$$\mathbb{P}_{\text{near}} := \mathbb{P} \backslash \mathbb{P}_{\text{far}} = \big\{(\sigma, \tau) \in \mathbb{P} \,\big|\, (\sigma, \tau) \text{ is inadmissible}\big\}. \tag{2.4}$$

The idea of the block partitioning $\mathbb{P}$ is as follows: For admissible blocks $(\sigma, \tau) \in \mathbb{P}_{\text{far}}$ we will compute the submatrix $A|_{\sigma \times \tau}$ approximately, e.g., by interpolation of the kernel. For inadmissible blocks $(\sigma, \tau) \in \mathbb{P}_{\text{near}}$, we will compute the submatrix $A|_{\sigma \times \tau}$ exactly.

With a given admissibility condition, e.g. (2.2), the following algorithm computes a block partitioning from a cluster tree $\mathbb{T}$. The recursive function is called with

`CreateBlockPartitioning(`$\emptyset$`,`$\emptyset$`,`$\mathcal{I}$`,`$\mathcal{J}$`)`.

Please note that Algorithm 2.3 is not used in praxis to build and store the block partitioning $\mathbb{P}$. Instead, $\mathbb{P}$ is built on the fly from the cluster tree and the admissibility condition, i.e. the following algorithm is somehow a built-in part in each of the subsequent algorithms. We refer, for instance, to the matrix-vector multiplication with $\mathcal{H}$-matrices in Algorithm 2.6.

Algorithm 2.3 (Create Block Partitioning (First Version)).

```
function CreateBlockPartitioning(var P_near,var P_far,σ,τ)
if (σ,τ) admissible
   add (σ,τ) to P_far
elseif sons(σ) ≠ ∅ and sons(τ) ≠ ∅
   for all (σ',τ') ∈ sons(σ) × sons(τ)
      call CreateBlockPartitioning(P_near,P_far,σ',τ')
else /* inadmissible block which cannot be split any further */
   add (σ,τ) to P_near
end
```

Definition. According to the block partitioning algorithm, there holds $\operatorname{level}(\sigma) = \operatorname{level}(\tau)$ for $(\sigma, \tau) \in \mathbb{P}$. Therefore, we can define the **depth of $\mathbb{P}$** by $\operatorname{depth}(\mathbb{P}) = \max\limits_{(\sigma,\tau)\in\mathbb{P}} \operatorname{level}(\sigma)$.

Remark. (i) For the fast multipole method (resp. $\mathcal{H}^2$-matrices) we will have to change the admissibility condition (2.2). Moreover, we will do some refinements of Algorithm 2.3. Details are postponed to Chapter 5.
(ii) In fact, a block partitioning is nothing but a quad tree, but we omit any further definition. We will use this observation, however, for the $\mathcal{H}$-arithmetics, which is the topic of Chapter 3.□

For a sparse matrix $A \in \mathbb{K}^{n\times n}$ a measure for the sparsity is the maximal number of non-zero entries per row and column, i.e. one considers

$$C_{\text{row}} = \max_{j=1,\dots,n} \big|\{k \,\big|\, A_{jk} \neq 0\}\big|, \quad \text{and} \quad C_{\text{col}} = \max_{k=1,\dots,n} \big|\{j \,\big|\, A_{jk} \neq 0\}\big|,$$

as well as the maximum of both, $C := \max\{C_{\text{row}}, C_{\text{col}}\}$. The idea of this definition carries over to the block partitioning, where we don't want too many blocks in our block-matrix.

Definition. Let $\mathbb{P}$ be a given block partitioning of $\mathcal{I} \times \mathcal{J}$. Then, we define the row- and column- **sparsity constants**

$$C_{\text{sparse}}^{\text{row}} := \max_{\sigma\in\mathbb{T}_{\mathcal{I}}} \big|\big\{\tau \in \mathbb{T}_{\mathcal{J}} \,\big|\, (\sigma,\tau) \in \mathbb{P}\big\}\big| \quad \text{and} \quad C_{\text{sparse}}^{\text{col}} := \max_{\tau\in\mathbb{T}_{\mathcal{J}}} \big|\big\{\sigma \in \mathbb{T}_{\mathcal{I}} \,\big|\, (\sigma,\tau) \in \mathbb{P}\big\}\big|. \tag{2.5}$$

Moreover, the **total sparsity constant** is $C_{\text{sparse}} := \max\{C_{\text{sparse}}^{\text{row}}, C_{\text{sparse}}^{\text{col}}\}$. Note that C_{sparse} is somehow a measure for the sparsity of the block partition.

The sparsity constant enters all subsequent complexity estimates for hierarchical matrices. Thus, it is important to observe *a posteriori* that C_{sparse} does not blow up, whenever we are dealing with mesh-refinements (as we will usually do): If C_{sparse} blows up, so does the complexity of arithmetic operations with hierarchical matrices as well as the storage requirements.

Lemma 2.4. *According to the definition and with $N = m + n$, there holds*

$$|\mathbb{P}| = |\mathbb{P}_{\text{near}}| + |\mathbb{P}_{\text{far}}| \leq \min\left\{C_{\text{sparse}}^{\text{row}}|\mathbb{T}_{\mathcal{I}}|\,,\, C_{\text{sparse}}^{\text{col}}|\mathbb{T}_{\mathcal{J}}|\right\} = \mathcal{O}(N) \tag{2.6}$$

as well as

$$\operatorname{depth}(\mathbb{P}) \leq \min\{\operatorname{depth}(\mathbb{T}_{\mathcal{I}}), \operatorname{depth}(\mathbb{T}_{\mathcal{J}})\} = \mathcal{O}(\log N) \tag{2.7}$$

as long as C_{sparse} stays uniformly bounded. ■

Remark. In the following, we will always "hide" the sparsity constant C_{sparse} in the Landau-$\mathcal{O}$ notation. In particular, complexity estimates for $N \to \infty$ only hold as long as C_{sparse} remains bounded. Moreover, we shall always assume that (at least one of) the trees $\mathbb{T}_{\mathcal{I}}$ and $\mathbb{T}_{\mathcal{J}}$ are generic and *not degenerate* so that $\operatorname{depth}(\mathbb{P}) = \mathcal{O}(\log N)$. □

2.3 $\mathcal{H}$-Matrices (19.05.2009)

Having defined cluster tree, admissibility condition, and block partitioning, we can finally touch the heart of the matter: hierarchical matrices. Throughout, we make the following assumptions:

- $\mathbb{T}_{\mathcal{I}}$ is a cluster tree for $\mathcal{I} = \{1,\dots,m\}$,
- $\mathbb{T}_{\mathcal{J}}$ is a cluster tree for $\mathcal{J} = \{1,\dots,n\}$,
- $\mathbb{A} : \mathbb{T}_{\mathcal{I}} \times \mathbb{T}_{\mathcal{J}} \to \{\texttt{admissible}, \texttt{nonadmissible}\}$ is a given admissibility condition, e.g. (2.2),

- $\mathbb{P}$ is the block partitioning induced by Algorithm 2.3.

For simplicity, we assume that the constants C_{leaf} for $\mathbb{T}_{\mathcal{I}}$ and $\mathbb{T}_{\mathcal{J}}$ coincide.

2.3.1 Definition of $\mathcal{H}$-Matrices

Definition. Let $p \in \mathbb{N}$ be the **rank of admissible blocks**. A matrix $A \in \mathbb{K}^{m\times n}$ is called **hierarchical matrix**, shortly **$\mathcal{H}$-matrix**, with respect to $\mathbb{T}_{\mathcal{I}} \times \mathbb{T}_{\mathcal{J}}$, $\mathbb{A}$, and p provided that there holds the following: For any admissible block $(\sigma,\tau) \in \mathbb{P}_{\text{far}}$, there are matrices $V_{\sigma\tau} \in \mathbb{K}^{|\sigma|\times p}$ and $W_{\sigma\tau} \in \mathbb{K}^{|\tau|\times p}$ such that

$$A|_{\sigma\times\tau} = V_{\sigma\tau} W_{\sigma\tau}^H. \tag{2.8}$$

We shall then write $A \in \mathcal{H}(p) := \mathcal{H}(\mathbb{T}_{\mathcal{I}}, \mathbb{T}_{\mathcal{J}}, \mathbb{A}, p)$.

Remark. For a hierarchical matrix A one only stores the near field $A|_{\sigma\times\tau}$ for $(\sigma,\tau) \in \mathbb{P}_{\text{near}}$ and the far field matrices $V_{\sigma\tau}, W_{\sigma\tau}$ for $(\sigma,\tau) \in \mathbb{P}_{\text{far}}$, i.e., the matrix is stored —as far as possible— in factorized form. □

Please note that, for fixed $\mathbb{T}_{\mathcal{I}}$, $\mathbb{T}_{\mathcal{J}}$, $\mathbb{P}$, and p, the set of all $\mathcal{H}$-matrices is neither closed with respect to addition nor with respect to matrix-matrix multiplication. For instance, note that the sum

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

of two rank-1-matrices can have rank 2. Nevertheless, one can define *approximate arithmetic operations*, called **$\mathcal{H}$-arithmetics**, which will be the topic of later sections. As a warm-up, we only consider the storage requirements (and thus the complexity for building the matrix) and the arithmetic complexity for the matrix-vector multiplication, where we stress that the latter is exact (instead of only approximate).

2.3.2 Storage Requirements for $\mathcal{H}$-Matrices

Proposition 2.5. *The storage requirements N_{Storage} for an $\mathcal{H}$-matrix $A \in \mathcal{H}(p)$ satisfy*

$$N_{\text{Storage}} \le C_{\text{sparse}}(\text{depth}(\mathbb{P}) + 1)\max\{C_{\text{leaf}}, p\}(m+n) = \mathcal{O}(pN\log N),$$

where $N = m + n$.

Proof. For $(\sigma,\tau) \in \mathbb{P}_{\text{near}}$, there holds either $\sigma \in \text{leaves}(\mathbb{T}_{\mathcal{I}})$ or $\tau \in \text{leaves}(\mathbb{T}_{\mathcal{J}})$ according to Algorithm 2.3. From the definition of a cluster tree, we therefore obtain either $|\sigma| \le C_{\text{leaf}}$ or $|\tau| \le C_{\text{leaf}}$, whence $|\sigma||\tau| \le C_{\text{leaf}}(|\sigma| + |\tau|)$. Together with the storage requirements $p(|\sigma| + |\tau|)$ for $(\sigma,\tau) \in \mathbb{P}_{\text{far}}$, this yields

$$N_{\text{Storage}} = \sum_{(\sigma,\tau)\in\mathbb{P}_{\text{far}}} p(|\sigma| + |\tau|) + \sum_{(\sigma,\tau)\in\mathbb{P}_{\text{near}}} |\sigma||\tau| \le \max\{C_{\text{leaf}}, p\} \sum_{(\sigma,\tau)\in\mathbb{P}} (|\sigma| + |\tau|)$$

The σ-part of the last sum is rewritten as follows:

$$\sum_{(\sigma,\tau)\in\mathbb{P}} |\sigma| = \sum_{\ell=0}^{\text{depth}(\mathbb{P})} \sum_{\substack{\sigma\in\mathbb{T}_{\mathcal{I}} \\ \text{level}(\sigma)=\ell}} \sum_{\substack{\tau\in\mathbb{T}_{\mathcal{J}} \\ (\sigma,\tau)\in\mathbb{P}}} |\sigma| \le C_{\text{sparse}} \sum_{\ell=0}^{\text{depth}(\mathbb{P})} \sum_{\substack{\sigma\in\mathbb{T}_{\mathcal{I}} \\ \text{level}(\sigma)=\ell}} |\sigma|$$

Note that on each level ℓ of the cluster tree, there holds

$$\sum_{\substack{\sigma\in\mathbb{T}_{\mathcal{I}}\\ \mathrm{level}(\sigma)=\ell}} |\sigma| \leq |\mathcal{I}| = n$$

according to the disjoint splitting in the tree structure. This implies

$$\sum_{(\sigma,\tau)\in\mathbb{P}} |\sigma| \leq C_{\mathrm{sparse}}(\mathrm{depth}(\mathbb{P})+1)n.$$

The same arguments for the τ-part of the sum conclude the proof. ■

Remark. The proof of Proposition 2.5 provides the estimate

$$\sum_{(\sigma,\tau)\in\mathbb{P}} (|\sigma|+|\tau|) \leq C_{\mathrm{sparse}}(\mathrm{depth}(\mathbb{P})+1)(m+n). \tag{2.9}$$

Throughout the lecture, this observation will be important, whenever we aim to provide complexity estimates. □

2.3.3 Matrix-Vector Multiplication with $\mathcal{H}$-Matrices

For any block matrix $A \in \mathbb{K}^{m\times n}$, there obviously holds

$$(Ax)_j = \sum_{\substack{(\sigma,\tau)\in\mathbb{P}\\ j\in\sigma}} \left(A|_{\sigma\times\tau}x|_\tau\right)_j$$

with $1 \leq j \leq m$. The following recursive algorithm is called with

`HMatrixVectorMultiplication(`y`,`$\mathcal{I}$`,`$\mathcal{J}$`,`A`,`x`)`

and computes the vector $y = Ax$. We remark that Algorithm 2.3 is part of Algorithm 2.6, i.e., the block partitioning is built on the fly. Note that the vector $y \in \mathbb{K}^m$ has to be initialized by 0.

Algorithm 2.6 (MVM with $\mathcal{H}$-Matrices).

```
function HMatrixVectorMultiplication(var y,σ,τ,A,x)
if (σ,τ) is admissible
   y|_σ := y|_σ + V_στ W_στ^H x|_τ
elseif sons(σ) ≠ ∅ and sons(τ) ≠ ∅
   for all (σ',τ') ∈ sons(σ) × sons(τ)
      call HMatrixVectorMultiplication(y,σ',τ',A,x)
else /* inadmissible block which cannot be split any further */
   y|_σ := y|_σ + A|_σ×τ x|_τ
end
```

Please note that, for $(\sigma,\tau) \in \mathbb{P}_{\mathrm{far}}$, we perform $V_{\sigma\tau}W_{\sigma\tau}^H x|_\tau$ in two steps: First, we compute $z = W_{\sigma\tau}^H x|_\tau$, second, we compute $V_{\sigma\tau}z$. With this convention, there holds the following proposition:

Proposition 2.7. *The number N_{MVM} of arithmetic operations for the matrix-vector multiplication with an $\mathcal{H}$-matrix $A \in \mathcal{H}(p)$ satisfies*

$$N_{\mathrm{Storage}} \leq N_{\mathrm{MVM}} \leq 2N_{\mathrm{Storage}},$$

with N_{Storage} from Proposition 2.5.

Proof. For a matrix $B \in \mathbb{K}^{M\times N}$, the matrix-vector multiplication with a vector $z \in \mathbb{K}^N$ needs N multiplications and $N-1$ additions to compute $(Bz)_j$. Altogether, one needs $M(2N-1)$ arithmetic operations to compute $Bz \in \mathbb{K}^M$. Hence, Algorithm 2.6 needs $|\sigma|(2|\tau|-1)$ operations for the matrix-vector multiplication per near field block $(\sigma,\tau) \in \mathbb{P}_{\mathrm{near}}$. Moreover, we need $|\sigma|$ additions to update $y|_\sigma$. In comparison, the storage requirements for a near field block are $|\sigma||\tau|$, and there holds

$$|\sigma||\tau| \leq 2|\sigma||\tau| = |\sigma|(2|\tau|-1) + |\sigma|$$

Furthermore, Algorithm 2.6 needs $|\sigma|(2p-1)+p(2|\tau|-1)$ operations per far field block $(\sigma,\tau) \in \mathbb{P}_{\mathrm{far}}$. Again we have to account for $|\sigma|$ additions to update $y|_\sigma$ in the end. In contrast to this, the storage requirements for a far field block are $p(|\sigma|+|\tau|)$. There holds

$$p(|\sigma|+|\tau|) = |\sigma|p + p|\tau| \leq |\sigma|(2p-1) + p(2|\tau|-1) + |\sigma| \leq 2p(|\sigma|+|\tau|).$$

From these two estimates, we conclude the proof if we sum the contributions over all far field and near field blocks of $\mathbb{P}$. ■

2.4 Tensorial Interpolation (17.03.2009)

During this section, let $p \in \mathbb{N}$ denote a *fixed* interpolation order. The task is to define a tensorial interpolation operator on $\mathbb{R}^d$. We start with the Lagrange interpolation problem on an interval $[a,b] \subset \mathbb{R}$.

2.4.1 Lagrange Interpolation on an Interval $[a,b]$

For a fixed order $p \in \mathbb{N}$, let $t_j \in [a,b]$ be p pairwise different interpolation nodes. Then, the **Lagrange interpolation problem** reads as follows: Given function values $y_j \in \mathbb{R}$, find a polynomial $q \in \mathcal{P}^{p-1}$ such that

$$q(t_j) = y_j \quad \text{for all } j = 1,\dots,p. \tag{2.10}$$

To prove that this problem has a unique solution q, we define the **Lagrange polynomials** associated with the node vector $(t_1,\dots,t_p)$ by

$$L_j(t) = \prod_{\substack{k=1\\ j\neq k}}^{p} \frac{t-t_k}{t_j-t_k}. \tag{2.11}$$

Obviously there holds $L_j \in \mathcal{P}^{p-1}$ and $L_j(t_k) = \delta_{jk}$ with Kronecker's delta.

Proposition 2.8. *For each vector $y \in \mathbb{K}^p$, there is a unique polynomial $q \in \mathcal{P}^{p-1}$ which solves (2.10). There holds $q = \sum_{j=1}^p y_j L_j$.*

Proof. According to the Kronecker property of the Lagrange polynomials, the polynomial $q = \sum_{j=1}^{p} y_j L_j$ solves (2.10). We consider the linear operator $T : \mathcal{P}^{p-1} \to \mathbb{K}^p$, $Tq = (q(t_1), \ldots, q(t_p))$ which evaluates a polynomial at the nodes t_j. The solvability of (2.10) means that T is surjective. From linearity and $\dim \mathcal{P}^{p-1} = p$, we infer that T is an isomorphism. Hence, the solution of (2.10) is unique. ■

Finally, we may define the interpolation operator $I_p : \mathcal{C}[a,b] \to \mathcal{P}^{p-1}$ by

$$I_p u = \sum_{j=1}^{p} u(t_j) L_j. \tag{2.12}$$

From the last proposition, we infer that, given $u \in \mathcal{C}[a,b]$ and $y_j = u(t_j)$, the solution of (2.10) reads $q = I_p u$. The following proposition concludes the main properties of the interpolation operator I_p.

Proposition 2.9. (i) *I_p is a projection onto $\mathcal{P}^{p-1}$, i.e., $I_p q = q$ for all $q \in \mathcal{P}^{p-1}$.*
(ii) *$I_p : \mathcal{C}[a,b] \to \mathcal{P}^{p-1}$ is linear and continuous with operator norm*

$$1 \le \|I_p\| := \sup_{u \in \mathcal{C}[a,b]} \frac{\|I_p u\|_{\infty,[a,b]}}{\|u\|_{\infty,[a,b]}} \le \sup_{t \in [a,b]} \sum_{j=1}^{p} |L_j(t)| < \infty. \tag{2.13}$$

(iii) *I_p has the approximation property*

$$\|u - I_p u\|_{\infty,[a,b]} \le 2 \frac{\|u^{(p)}\|_{\infty,[a,b]}}{p!} \max_{t \in [a,b]} \prod_{j=1}^{p} |t - t_j| \quad \text{for all } u \in \mathcal{C}^p[a,b]. \tag{2.14}$$

Proof. (i) follows from the uniqueness of the interpolating polynomial since both polynomials $q, I_p q \in \mathcal{P}^{p-1}$ have the same function values at the nodes t_j. (ii) Obviously the operator I_p is linear. It remains to prove the estimate (2.13). Here, the lower bound follows from the projection property $I_p q = q$ for all $q \in \mathcal{P}^{p-1}$. The upper bound follows from the triangle inequality. (iii) We first consider the case $f : [a,b] \to \mathbb{R}$. For fixed $y \in [a,b]$, one considers the function

$$f(t) := (u - I_p u)(t)\omega(y) - (u - I_p u)(y)\omega(t), \quad \text{where } \omega(t) := \prod_{j=1}^{p} (t - t_j).$$

Note that f has $p+1$ zeros in $[a,b]$. By induction, Rolle's theorem provides a zero $\zeta \in (a,b)$ of $f^{(p)}$. By definition of f, we obtain

$$0 = f^{(p)}(\zeta) = u^{(p)}(\zeta)\omega(y) - (u - I_p u)(y)\, p!$$

which implies

$$(u - I_p u)(y) = \frac{u^{(p)}(\zeta)}{p!}\omega(y)$$

Considering the absolute values and taking the supremum over $\zeta = \zeta(y)$ and y, we conclude the proof for $\mathbb{K} = \mathbb{R}$ — even without the factor 2 in estimate (2.14). For $\mathbb{K} = \mathbb{C}$, we may apply the estimate for $\operatorname{Re} f$ and $\operatorname{Im} f$ separately. ■

One may expect that the upper bound for the operator norm grows with increasing order $p \to \infty$. In the following sections, we consider a special choice of the node vector which leads to a slow (logarithmic) growth.

2.4.2 Čebyšev Interpolation on the Reference Interval $[-1,1]$

We consider the reference interval $[-1,1]$ and define the **Čebyšev nodes** $\zeta_j \in [-1,1]$ by

$$\zeta_j = \cos\Big(\frac{2j-1}{p}\,\frac{\pi}{2}\Big) \quad \text{for } j = 1,\dots,p,$$

which are just the nodes are the zeros of the p-th Čebyšev polynomial. For the corresponding Lagrange polynomials, we write L_p^{ref}. The induced interpolation operator is denoted by I_p^{ref} and is called **Čebyšev interpolation operator**. The following theorem is an improvement of the stability estimate (2.11) and the approximation estimate (2.13) which uses the special choice of the interpolation nodes.

Theorem 2.10. (i) *The operator norm Λ_p of I_p^{ref} satisfies*

$$1 \le \Lambda_p = \frac{1}{p}\sum_{j=1}^{p} \cot\frac{(2j-1)\pi}{4p} \le \frac{2}{\pi}\log(p) + 1 \le p \tag{2.15}$$

(ii) *The approximation estimate for I_p^{ref} reads*

$$\|u - I_p^{\mathrm{ref}} u\|_{\infty,[-1,1]} \le 4\,\frac{2^{-p}}{p!}\,\|u^{(p)}\|_{\infty,[-1,1]} \quad \textit{for all } u \in \mathcal{C}^p[-1,1]. \tag{2.16}$$

***Proof of Theorem 2.10* (ii).** For $p \in \mathbb{N}$, we define the **Čebyšev polynomial**

$$C_p(\zeta) := \cos(p \arccos\zeta) \quad \text{for } \zeta \in [-1,1]. \tag{2.17}$$

For $\phi \in [0,\pi]$ and $\zeta = \cos\phi$, there holds $C_p(\zeta) = \cos(p\phi)$. Thus, the Čebyšev nodes are just the zeros of C_p in $[-1,1]$. The addition formula for the cosine reads

$$\cos(x) + \cos(y) = 2\cos\Big(\frac{x+y}{2}\Big)\cos\Big(\frac{x-y}{2}\Big) \quad \text{for all } x,y \in \mathbb{R}.$$

Plugging in $x = (p+1)\phi$ and $y = (p-1)\phi$, we obtain the recursion formula

$$2\zeta C_p(\zeta) - C_{p-1}(\zeta) = 2\cos(\phi)\cos(p\phi) - \cos((p-1)\phi) = \cos((p+1)\phi) = C_{p+1}(\zeta). \tag{2.18}$$

From induction, we derive that C_p is in fact a polynomial of degree p on $[-1,1]$ with leading coefficient 2^{p-1}. Therefore, the definition of C_p implies

$$1 = \|C_p\|_{\infty,[-1,1]} = 2^{p-1} \max_{\zeta\in[-1,1]} \prod_{j=1}^{p} |\zeta - \zeta_j|.$$

Plugging this into (2.14), we prove (2.16). ∎

For the proof of Theorem 2.10 (i), we refer to the monograph of DeVore and Lorentz.

2.4.3 Čebyšev Interpolation on an Interval $[a, b]$

We map the Čebyšev nodes from $[-1, 1]$ onto an arbitrary interval $[a, b]$ by use of the **affine transformation**

$$\psi(\zeta) = \frac{1}{2}\big(a + b + \zeta(b - a)\big),$$

i.e. we define the **Čebyšev nodes** on $[a, b]$ by

$$t_j = \psi(\zeta_j) \quad \text{for } j = 1, \dots, p. \tag{2.19}$$

Now, let $L_j \in \mathcal{P}^{p-1}$ denote the corresponding Lagrange polynomials and I_p the induced interpolation operator. With the bijectivity of ψ, one can rewrite the estimates (2.15)–(2.16) on the interval $[a, b]$.

Corollary 2.11. (i) *The operator norm of I_p is Λ_p, i.e.*

$$\|I_p u\|_{\infty,[a,b]} \le \Lambda_p \, \|u\|_{\infty,[a,b]} \quad \textit{for all } u \in \mathcal{C}[a, b]. \tag{2.20}$$

(ii) *The approximation estimate for I_p reads*

$$\|u - I_p u\|_{\infty,[a,b]} \le 4 \, \frac{4^{-p}}{p!} \, (b - a)^p \|u^{(p)}\|_{\infty,[a,b]} \quad \textit{for all } u \in \mathcal{C}^p[a, b]. \tag{2.21}$$

Proof. (i) follows from the fact that, for all $v \in \mathcal{C}[a, b]$,

$$\|v\|_{\infty,[a,b]} = \sup_{t\in[a,b]} |v(x)| = \sup_{\zeta\in[-1,1]} |v(\psi(\zeta))| = \|v \circ \psi\|_{\infty,[-1,1]}.$$

The proof of (ii) additionally needs, that $(u \circ \psi)^{(p)}(\zeta) = \Big(\frac{b-a}{2}\Big)^p u^{(p)}(\psi(\zeta))$. ■

2.4.4 Tensorial Čebyšev Interpolation on a compact box in $\mathbb{R}^d$

We consider the compact (bounding) box $B := \prod_{j=1}^{d} [a_j, b_j] \subseteq \mathbb{R}^d$ and define an interpolation operator $\mathcal{I}_p$ by tensorial interpolation: To each $1 \le j \le p^d$, there corresponds a unique vector $(j_1, \dots, j_d) \in \{1, \dots, p\}^d$. We define the tensor interpolation nodes

$$\mathbf{t}_j = (t_{j_1}^{(1)}, \dots, t_{j_d}^{(d)})$$

with $t_m^{(n)}$ denoting the m-th Čebyšev node in the interval $[a_n, b_n]$. Now let $L_{j_m}^{(m)} \in \mathcal{P}^{p-1}$ denote the Lagrange polynomial corresponding to t_{j_m}. The tensorial Lagrange polynomials are given by

$$\mathcal{L}_j(x) = \prod_{m=1}^{d} L_{j_m}^{(m)}(x_m),$$

and there obviously holds $\mathcal{L}_j(\mathbf{t}_k) = \delta_{jk}$. Finally, we have defined an interpolation operator

$$\mathcal{I}_p u = \sum_{j=0}^{p^d} u(x_j)\mathcal{L}_j \quad \text{for } u \in \mathcal{C}(B).$$

From the 1D case, we now derive the following result:

Theorem 2.12. (i) *The operator $\mathcal{I}_p$ is a projection, i.e.*

$$\mathcal{I}_p(\mathcal{I}_p u) = \mathcal{I}_p u \quad \text{for all } u \in \mathcal{C}(B). \tag{2.22}$$

(ii) *The operator $\mathcal{I}_p$ is linear and continuous with operator norm $\leq \Lambda_p^d$, i.e.*

$$1 \leq \|\mathcal{I}_p u\|_{\infty,B} \leq \Lambda_p^d \, \|u\|_{\infty,B} \quad \text{for all } u \in \mathcal{C}(B). \tag{2.23}$$

(iii) *The operator $\mathcal{I}_p$ has the approximation property*

$$\|u - \mathcal{I}_p u\|_{\infty,B} \leq 4 \, \frac{4^{-p}}{p!} \, \Lambda_p^{d-1} \operatorname{diam}(B)^p \sum_{j=1}^{d} \|\partial_j^p u\|_{\infty,B} \quad \text{for all } u \in \mathcal{C}^{p+1}(B) \tag{2.24}$$

Proof. (i) We define the interpolation $\mathcal{I}_p^{(j)} : \mathcal{C}(B) \to \mathcal{C}(B)$ in the j-th direction by

$$(\mathcal{I}_p^{(j)} u)(x) = \sum_{k=0}^{p} u(x_1, \ldots, x_{j-1}, t_k^{(j)}, x_{j+1}, \ldots, x_d) L_k^{(j)}(x_j)$$

We observe some basic facts: First, the operators $\mathcal{I}_p^{(j)}$ commute, i.e. $\mathcal{I}_p^{(j)} \mathcal{I}_p^{(k)} = \mathcal{I}_p^{(k)} \mathcal{I}_p^{(j)}$ for all $1 \leq j, k \leq d$. Second, there holds $\mathcal{I}_p = \prod_{j=1}^{d} \mathcal{I}_p^{(j)}$ in the sense of the composition. Third, the operator $\mathcal{I}_p^{(j)}$ satisfies the projection property $\mathcal{I}_p^{(j)} \mathcal{I}_p^{(j)} = \mathcal{I}_p^{(j)}$ due to Proposition 2.9 (i). From this we conclude $\mathcal{I}_p \mathcal{I}_p = \mathcal{I}_p$. Moreover, the operator $\mathcal{I}_p^{(j)}$ inherits the stability

$$\|\mathcal{I}_p^{(j)} v\|_\infty \leq \Lambda_p \|v\|_\infty \quad \text{for } v \in \mathcal{C}(B) \tag{2.25}$$

and the approximation property

$$\|v - \mathcal{I}_p^{(j)}\|_\infty \leq 4 \, \frac{4^{-p}}{p!} \, (b_j - a_j)^p \|\partial_j^p v\|_\infty \leq 4 \frac{4^{-p}}{p!} \operatorname{diam}(B)^p \, \|\partial_j^p v\|_\infty \quad \text{for } v \in \mathcal{C}^p(B) \tag{2.26}$$

from the one dimensional case. (ii) follows immediately from

$$\|\mathcal{I}_p u\|_\infty \leq \prod_{j=1}^{d} \|\mathcal{I}_p^{(j)}\| \|u\|_\infty \leq \Lambda_p^d \|u\|_\infty.$$

(iii) Moreover, with a telescope series we write

$$u - \mathcal{I}_p u = u - \Big(\prod_{j=1}^{d} \mathcal{I}_p^{(j)} \Big) u = \sum_{k=1}^{d} \Big(\prod_{j=1}^{k-1} \mathcal{I}_p^{(j)} u - \prod_{j=1}^{k} \mathcal{I}_p^{(j)} u \Big) = \sum_{k=1}^{d} \Big(\prod_{j=1}^{k-1} \mathcal{I}_p^{(j)} \Big) (u - \mathcal{I}_p^{(k)} u).$$

By use of (2.25)–(2.26), this leads to

$$\begin{aligned} \|u - \mathcal{I}_p u\|_\infty \leq \sum_{k=1}^{d} \Big\| \Big(\prod_{j=1}^{k-1} \mathcal{I}_p^{(j)} \Big) (u - \mathcal{I}_p^{(k)} u) \Big\|_\infty &\leq \sum_{k=1}^{d} \Lambda_p^{k-1} \|u - \mathcal{I}_p^{(k)} u\|_\infty \\ &\leq 4 \, \frac{4^{-p}}{p!} \operatorname{diam}(B)^p \sum_{k=1}^{d} \Lambda_p^{k-1} \|\partial_k^p u\|_\infty \end{aligned}$$

which concludes the proof. ∎

2.5 Asymptotically Smooth Kernels (17.03.2009)

Definition. A (possibly complex valued) kernel function $\kappa(x,y)$ is **asymptotically smooth** provided if is smooth for $x \neq y$ and provided there are constants $c_1, c_2 > 0$ and the **order of singularity** $s \in \mathbb{R}$ such that

$$|\partial_x^\alpha \partial_y^\beta \kappa(x,y)| \leq c_1 \, (c_2|x-y|)^{-(|\alpha|+|\beta|+s)} (\alpha+\beta)! \tag{2.27}$$

for all multi-indices $\alpha, \beta \in \mathbb{N}_0^d$ with $|\alpha| + |\beta| \geq 1$.

Exercise. (i) The kernel $\kappa(x,y) = |x-y|^{-s}$ is asymptotically smooth with $c_1 = 1 = c_2$.
(ii) The kernel $\kappa(x,y) = \log|x-y|$ is asymptotically smooth with $c_1 = c_2 = 1$ and $s = 0$.
(iii) Any partial derivative $\widetilde{\kappa}(x,y)$ of an asymptotically smooth kernel $\kappa(x,y)$ is again asymptotically smooth with constants $\widetilde{c_2} = c_2 + \varepsilon$ and $\widetilde{c_1} = \widetilde{c_1}(c_1, c_2, \varepsilon)$ for any arbitrary $\varepsilon > 0$. □

Theorem 2.13. *Let κ be an asymptotically smooth kernel and B_σ, B_τ non-degenerate boxes in $\mathbb{R}^d$ with $\operatorname{diam}(B_\tau) \leq \eta \operatorname{dist}(B_\sigma, B_\tau)$ for some fixed $\eta > 0$. If $\mathcal{I}_p^\tau$ denotes the tensorial Čebyšev interpolation in y-direction with tensorial nodes in B_τ, there holds*

$$\|\kappa - \mathcal{I}_p^\tau \kappa\|_{\infty, B_\sigma \times B_\tau} \leq 4dc_1 (\eta/c_2)^s \operatorname{diam}(B_\tau)^{-s} \Lambda_p^{d-1} \left(\frac{\eta}{4c_2}\right)^p \tag{2.28}$$

i.e. we obtain exponential convergence with respect to p provided $\eta < 4c_2$ since Λ_p grows only logarithmically.

Proof. With Theorem 2.12, there holds

$$\begin{aligned}
\|\kappa - \mathcal{I}_p^\tau \kappa\|_{\infty, B_\sigma \times B_\tau} &\leq 4 \frac{4^{-p}}{p!} \Lambda_p^{d-1} \operatorname{diam}(B_\tau)^p \sum_{j=1}^d \|\partial_{y,j}^p \kappa\|_{\infty, B_\sigma \times B_\tau} \\
&\leq 4 \frac{4^{-p}}{p!} \Lambda_p^{d-1} \operatorname{diam}(B_\tau)^p \left[dc_1 \big(c_2 \operatorname{dist}(B_\sigma, B_\tau)\big)^{-(p+s)} p! \right] \\
&\leq 4dc_1 (\eta/c_2)^s \operatorname{diam}(B_\tau)^{-s} \Lambda_p^{d-1} \Big(\frac{\eta}{4c_2}\Big)^p,
\end{aligned}$$

where we have used that $\frac{\operatorname{diam}(B_\tau)}{\operatorname{dist}(B_\sigma, B_\tau)} \leq \eta$. ■

Remark. It is a result of Melenk that (2.28) can be improved as follows: There holds

$$\|\kappa - \mathcal{I}_p^\tau \kappa\|_{\infty, B_\sigma \times B_\tau} \leq C \, p \Lambda_p^{d-1} \, (1 + 2c_2/\eta)^{-p},$$

where $C > 0$ is a constant that depends on B_σ, B_τ, and κ but not on p. In particular, this proves exponential decay with $p \to \infty$ without any restrictions on the choice of η. □

2.6 $\mathcal{H}$-Matrices by Interpolation (17.03.2009)

Finally, we want to analyze the approximation error $\|A - A_p\|$, where $A \in \mathbb{K}^{m \times n}$ is a dense matrix and A_p is an $\mathcal{H}$-matrix approximant obtained from interpolation.

Assumption on Matrix Entries. We assume that the matrix entries have a special structure, namely that A satisfies

$$A_{jk} = \kappa(x_j, y_k) \quad \text{for all } j = 1, \dots, m \text{ and } k = 1, \dots, n \tag{A1}$$

with given evaluation points $x_j, y_k \in \mathbb{R}^d$ or

$$A_{jk} = \int_\Omega \kappa(x_j, y)\psi_k(y)\, dy \quad \text{for all } j = 1, \dots, m \text{ and } k = 1, \dots, n \tag{A2}$$

with given evaluation points $x_j \in \mathbb{R}^d$ and functions $\psi_k \in L^1(\Omega)$ or

$$A_{jk} = \int_\omega \int_\Omega \phi_j(x)\kappa(x, y)\psi_k(y)\, dy\, dx \quad \text{for all } j = 1, \dots, m \text{ and } k = 1, \dots, n \tag{A3}$$

with given functions $\phi_j \in L^1(\omega)$ and $\psi_k \in L^1(\Omega)$. Here, $\kappa(x, y)$ is an asymptotically smooth kernel.

Assumption on Block Partitioning. We assume that $\mathbb{T}_\mathcal{I}$ and $\mathbb{T}_\mathcal{J}$ are cluster trees for the index sets $\mathcal{I} = \{1, \dots, m\}$ and $\mathcal{J} = \{1, \dots, n\}$. Here, the necessary geometrical data for the degrees of freedom is given as follows:

- For (A1), $\operatorname{supp}(j) = \{x_j\}$ and $\operatorname{supp}(k) = \{y_k\}$.
- For (A2), $\operatorname{supp}(j) = \{x_j\}$ and $\operatorname{supp}(k) = \operatorname{supp}(\psi_k)$.
- For (A3), $\operatorname{supp}(j) = \operatorname{supp}(\phi_j)$ and $\operatorname{supp}(k) = \operatorname{supp}(\psi_k)$.

Moreover, let $\mathbb{P}$ be the block partitioning induced by Algorithm 2.3, where $(\sigma, \tau) \in \mathbb{P}$ is called admissible provided that

$$\min\{\operatorname{diam}(B_\sigma), \operatorname{diam}(B_\tau)\} \le \eta \operatorname{dist}(B_\sigma, B_\tau). \tag{$\mathbb{A}$}$$

According to Theorem 2.13, we assume that the fixed parameter $\eta > 0$ satisfies $\eta < 4c_2$ with $c_2 > 0$ from the asymptotic smoothness of $\kappa(x, y)$.

Assembling the $\mathcal{H}$-Matrix. For each block $(\sigma, \tau) \in \mathbb{P}$, we have to assemble $A_p|_{\sigma\times\tau}$. To this end, we distinguish three cases:

(i) For $(\sigma, \tau) \in \mathbb{P}_{\text{near}}$, we define $A_p|_{\sigma\times\tau} := A|_{\sigma\times\tau}$.

(ii) For $(\sigma, \tau) \in \mathbb{P}_{\text{far}}$ and $\operatorname{diam}(B_\sigma) \le \operatorname{diam}(B_\tau)$, we approximate $A|_{\sigma\times\tau}$ as follows: We replace the kernel $\kappa(x, y)$ in the computation of A_{jk} for $(j, k) \in \sigma\times\tau$ by interpolation in x-direction

$$\kappa_p^{\sigma\tau}(x, y) := \mathcal{I}_p^\sigma \kappa(x, y) = \sum_{\ell=1}^{p^d} \kappa(x_\ell^\sigma, y)\mathcal{L}_\ell^\sigma(x).$$

For example, an entry A_{jk} corresponding to (A2) is approximated by

$$A_{jk} \approx (A_p)_{jk} := \int_\Omega \kappa_p^{\sigma\tau}(x_j, y)\psi_k(y)\, dy = \sum_{\ell=1}^{p^d} \mathcal{L}_\ell^\sigma(x_j) \int_\Omega \kappa(x_\ell^\sigma, y)\psi_k(y)\, dy.$$

Defining matrices $V_{\sigma\tau} \in \mathbb{K}^{|\sigma|\times p^d}$ and $W_{\sigma\tau} \in \mathbb{K}^{|\tau|\times p^d}$ by

$$(V_{\sigma\tau})_{j\ell} := \mathcal{L}_\ell^\sigma(x_j) \quad \text{and} \quad (W_{\sigma\tau})_{k\ell} := \int_\Omega \kappa(x_\ell^\sigma, y)\psi_k(y)\, dy,$$

we are led to

$$A|_{\sigma\times\tau} \approx A_p|_{\sigma\times\tau} = V_{\sigma\tau} W_{\sigma\tau}^H.$$

(iii) For $(\sigma,\tau) \in \mathbb{P}_{\text{far}}$ and $\operatorname{diam}(B_\tau) \le \operatorname{diam}(B_\sigma)$, we approximate $A|_{\sigma\times\tau}$ by replacing the kernel $\kappa(x,y)$ in the computation of A_{jk} for $(j,k) \in \sigma \times \tau$ by interpolation in y-direction

$$\kappa_p^{\sigma\tau}(x,y) := \mathcal{I}_p^\tau \kappa(x,y) = \sum_{\ell=1}^{p^d} \kappa(x, y_\ell^\tau)\mathcal{L}_\ell^\tau(y).$$

Again, we consider an entry A_{jk} in case of (A2). This becomes

$$A_{jk} \approx (A_p)_{jk} = \int_\Omega \kappa_p^{\sigma\tau}(x_j,y)\psi_k(y)\,dy = \sum_{\ell=1}^{p^d} \kappa(x_j,y_\ell^\tau) \int_\Omega \mathcal{L}_\ell^\tau(y)\psi_k(y)\,dy.$$

Defining matrices $V_{\sigma\tau} \in \mathbb{K}^{|\sigma|\times p^d}$ and $W_{\sigma\tau} \in \mathbb{K}^{|\tau|\times p^d}$ by

$$(V_{\sigma\tau})_{j\ell} := \kappa(x_j,y_\ell^\tau) \quad \text{and} \quad (W_{\sigma\tau})_{k\ell} := \int_\Omega \mathcal{L}_\ell^\tau(y)\psi_k(y)\,dy,$$

we are again led to a factorization

$$A|_{\sigma\times\tau} \approx A_p|_{\sigma\times\tau} = V_{\sigma\tau}W_{\sigma\tau}^H.$$

Altogether, we thus obtain a hierarchical matrix $A_p \in \mathcal{H}(\mathbb{T}_\mathcal{I}, \mathbb{T}_\mathcal{J}, \mathbb{A}, p^d)$ which approximates $A \in \mathbb{K}^{m\times n}$. We stress that we don't have to compute the exact matrix A in practice. We only compute the original entries for the near-field blocks.

Remark. Mathematically, the aforegoing definition of $V_{\sigma\tau}$ and $W_{\sigma\tau}$ is not correct since, for instance, the row indices of $V_{\sigma\tau}$ are $j' = 1,\dots,|\sigma|$ instead of $j \in \sigma$ as above. The correct definition would read as follows: For a cluster $\sigma \subseteq \mathcal{I}$, one fixes a numbering of the indices $j \in \sigma$, i.e, we define $\sigma' := \{1,\dots,|\sigma|\}$ and choose a bijection $\pi_\sigma : \sigma' \to \sigma$. For an index $j \in \sigma$, the *local index* $j' \in \sigma'$ is defined by $\pi_\sigma(j') = j$. Then, $V_{\sigma\tau} \in \mathbb{K}^{|\sigma|\times p^d}$ is defined by

$$(V_{\sigma\tau})_{j',\ell} := \mathcal{L}_\ell^\sigma(x_j) \quad \text{and} \quad (V_{\sigma\tau})_{j',\ell} := \kappa(x_j,y_\ell^\tau)$$

for case (ii) and (iii), respectively. However, this formalism has to be treated carefully in case of an implementation! □

Assumption on Interpolation Degree. In any of the cases (ii) and (iii) of the assembling step, Theorem 2.13 applies and proves, for all $(\sigma,\tau) \in \mathbb{P}_{\text{far}}$,

$$\|\kappa - \kappa_p^{\sigma\tau}\|_{\infty,B_\sigma\times B_\tau} \le 2dc_1(\eta/c_2)^s \min\{\operatorname{diam}(B_\sigma),\operatorname{diam}(B_\tau)\}^{-s}\,\Lambda_p^{d-1}\left(\frac{\eta}{4c_2}\right)^p. \tag{2.29}$$

Since the right-hand side tends to zero exponentially fast with the interpolation degree $p \to \infty$, we assume that there is a constant $C_p > 0$ such that

$$\|\kappa - \kappa_p^{\sigma\tau}\|_{\infty,B_\sigma\times B_\tau} \le C_p \quad \text{for all } (\sigma,\tau) \in \mathbb{P}_{\text{far}}, \tag{2.30}$$

where $C_p \xrightarrow{p\to\infty} 0$ exponentially fast.

Approximation Error. In the previous sections, we have seen that the storage requirements for A_p and the computational complexity for the matrix-vector multiplication are almost linear. In the remaining part of this section, we are dealing with a priori error estimates for $A - A_p$

with respect to the Frobenius norm as well as to the operator norm induced by the Euclidean norm.

Definition. For a matrix $A \in \mathbb{K}^{m\times n}$, the **Frobenius norm** reads

$$\|A\|_F := \Big(\sum_{j=1}^{m}\sum_{k=1}^{n} |A_{jk}|^2\Big)^{1/2}. \tag{2.31}$$

Moreover, we consider the **ℓ_2-operator norm** which is induced by the Euclidean norm

$$\|A\|_2 := \sup_{\substack{x\in\mathbb{K}^n\\ x\neq 0}} \frac{\|Ax\|_2}{\|x\|_2}. \tag{2.32}$$

The reader may want to check, that both, $\|\cdot\|_F$ and $\|\cdot\|_2$, are norms on $\mathbb{K}^{m\times n}$. Moreover, $\|\cdot\|_F$ is induced by a scalar product. Usually, we aim for error control with respect to $\|\cdot\|_2$ but $\|\cdot\|_F$ is more convenient for the analysis. Our first result states that the Frobenius norm provides an upper bound for $\|\cdot\|_2$.

Lemma 2.14. *For $A \in \mathbb{K}^{m\times n}$, there holds $\|A\|_2 \le \|A\|_F$.*

Proof. For $x \in \mathbb{K}^n$, the Cauchy inequality yields

$$\|Ax\|_2 = \Big(\sum_{j=1}^{m}\Big|\sum_{k=1}^{n} A_{jk}x_k\Big|^2\Big)^{1/2} \le \Big(\sum_{j=1}^{m}\Big[\sum_{k=1}^{n}|A_{jk}|^2\Big]\Big[\sum_{k=1}^{n}|x_k|^2\Big]\Big)^{1/2} = \|A\|_F\|x\|_2.$$

If we divide by $\|x\|_2$ and take the supremum, we prove $\|A\|_2 \le \|A\|_F$. ■

Proposition 2.15. *Let $A_p \in \mathcal{H}(\mathbb{T}_\mathcal{I}, \mathbb{T}_\mathcal{J}, \mathbb{A}, p^d)$ be the constructed $\mathcal{H}$-approximant of $A \in \mathbb{K}^{m\times n}$. Under the aforegoing assumptions, we have the following error estimates with respect to the Frobenius norm:*
(i) *If the entries of A are given by* (A1)*, there holds*

$$\|A - A_p\|_F \le C_p\sqrt{m}\sqrt{n}. \tag{2.33}$$

(ii) *If the entries of A are given by* (A2)*, there holds*

$$\|A - A_p\|_F \le C_p\sqrt{m}\sqrt{n}\max_{k=1,\dots,n}\|\psi_k\|_{L^1(\Omega)} \tag{2.34}$$

(iii) *If the entries of A are given by* (A3)*, there holds*

$$\|A - A_p\|_F \le C_p\sqrt{m}\sqrt{n}\max_{j=1,\dots,m}\|\phi_j\|_{L^1(\omega)}\max_{k=1,\dots,n}\|\psi_k\|_{L^1(\Omega)} \tag{2.35}$$

Proof. For $(j,k) \in \mathcal{I}\times\mathcal{J}$, there is a unique block $(\sigma,\tau) \in \mathbb{P}$ such that $(j,k) \in \sigma\times\tau$. For $(\sigma,\tau) \in \mathbb{P}_{\text{near}}$, there holds $A_{jk} = (A_p)_{jk}$. For $(\sigma,\tau) \in \mathbb{P}_{\text{far}}$, we have to consider the type of matrix entries: (i) In case of (A1), we derive from $x_j \in \cup\sigma \subseteq B_\sigma$ and $y_k \in \cup\tau \subseteq B_\tau$ that

$$\big|A_{jk} - (A_p)_{jk}\big| = \big|\kappa(x_j,y_k) - \kappa_p^{\sigma\tau}(x_j,y_k)\big| \le \|\kappa - \kappa_p^{\sigma\tau}\|_{\infty,B_\sigma\times B_\tau} \le C_p,$$

whence the Frobenius error reads

$$\|A - A_p\|_F^2 = \sum_{j,k} \big|A_{jk} - (A_p)_{jk}\big|^2 \leq C_p^2 mn.$$

(ii) In case of (A2), the Hölder inequality and $\operatorname{supp}(\psi_k) \subseteq \cup\tau \subseteq B_\tau$ prove

$$\begin{aligned}
\big|A_{jk} - (A_p)_{jk}\big| &= \Big| \int_\Omega \big(\kappa(x_j, y) - \kappa_p^{\sigma\tau}(x_j, y)\big)\psi_k(y)\, dy \Big| \\
&= \Big| \int_{B_\tau} \big(\kappa(x_j, y) - \kappa_p^{\sigma\tau}(x_j, y)\big)\psi_k(y)\, dy \Big| \\
&\leq \|\kappa(x_j, \cdot) - \kappa_p^{\sigma\tau}(x_j, \cdot)\|_{\infty, B_\tau} \|\psi_k\|_{L^1(B_\tau)} \\
&\leq C_p \max_{k=1,\dots,n} \|\psi_k\|_{L^1(\Omega)}.
\end{aligned}$$

Summing up this estimate, we prove (2.34). (iii) In case of (A1), the same technique yields

$$\begin{aligned}
\big|A_{jk} - (A_p)_{jk}\big| &= \Big| \int_{B_\sigma} \int_{B_\tau} \phi_j(x)\big(\kappa(x, y) - \kappa_p^{\sigma\tau}(x, y)\big)\psi_k(y)\, dy\, dx \Big| \\
&\leq \|\kappa - \kappa_p^{\sigma\tau}\|_{\infty, B_\sigma \times B_\tau} \|\phi_j\|_{L^1(B_\sigma)} \|\psi_k\|_{L^1(B_\tau)} \\
&\leq C_p \max_{j=1,\dots,m} \|\phi_j\|_{L^1(\omega)} \max_{k=1,\dots,n} \|\psi_k\|_{L^1(\Omega)},
\end{aligned}$$

which concludes the proof. ■

Remark. Note that we have only analyzed the error $\|A - A_p\|_F$ for fixed $A \in \mathbb{K}^{m\times n}$ and, in particular, fixed dimensions m, n. In practice, the dimensions $m, n \in \mathbb{N}$ usually grow along the computation, according to certain mesh-refinements. Then, the error $\|A - A_p\|_F$ might grow as well, due to the dependence of C_p on $\min\{\operatorname{diam}(B_\sigma), \operatorname{diam}(B_\tau)\}^{-s}$ for admissible blocks. In this case, one thus needs to increase p appropriately. For typical applications, one can show that $p = \mathcal{O}(\log(m + n))$ is sufficient. □

Remark. In many applications, the functions ϕ_j (as well as the function ψ_k) satisfy

- $|\operatorname{supp}(\phi_j)| \sim |\Omega|/m$,
- $|\phi_j| \leq 1$.

Therefore, we have $\|\phi_j\|_{L^1} = \mathcal{O}(1/m)$ which leads to an improvement of the error estimates in case of (A2) and (A3) if compared with (A1). □

Chapter 3

Black-Box $\mathcal{H}$-Algorithms

3.1 Singular Value Decomposition (19.05.2009)

This section provides some mathematical background to provide a deeper understanding of $\mathcal{H}$-matrices. For the moment, think of an admissible block $M = A|_{\sigma\times\tau}$ of an $\mathcal{H}$-matrix A. Then, M is given in factorized form $M = UV^H$, and $\operatorname{rank}(M) \leq p$ is usually much less than the dimensions of M.

3.1.1 Characterization of Low-Rank Matrices

The fundamental result is the following theorem on the existence of the singular value decomposition.

Theorem 3.1. *For any matrix $M \in \mathbb{K}^{m\times n}$, there are orthogonal matrices $U \in \mathbb{K}^{m\times m}$ and $V \in \mathbb{K}^{n\times n}$ and a unique matrix $\Sigma \in \mathbb{R}^{m\times n}$ such that*

- $M = U\Sigma V^H$,
- $\Sigma_{jk} = \sigma_j \delta_{jk}$,
- $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min\{m,n\}} \geq 0$.

This factorization is called **singular value decomposition**. *The numbers σ_j are called* **singular values**. ■

We stress some immediate consequences of Theorem 3.1:

Corollary 3.2. *Let $M \in \mathbb{K}^{m\times n}$ be an arbitrary matrix.*
(i) *The singular values σ_j of M are precisely the nonnegative roots of $M^H M$, i.e. $\sigma \geq 0$ is singular value of M if and only if σ^2 is eigenvalue of $M^H M$.*
(ii) *There holds* $\operatorname{rank}(M) \leq p$ *if and only if* $\sigma_{p+1} = \cdots = \sigma_{\min\{m,n\}} = 0$.
(iii) *There holds* $\operatorname{rank}(M) \leq p$ *if and only if there are matrices $V \in \mathbb{K}^{m\times p}$ and $W \in \mathbb{K}^{n\times p}$ such that $M = VW^H$.*

Proof. (i) The singular value decomposition $M = U\Sigma V^H$ implies

$$M^H M = V\Sigma^T \Sigma V^H.$$

From $V^H = V^{-1}$, we infer that $M^H M$ and $\Sigma^T\Sigma$ have the same eigenvalues. (ii) According to Linear Algebra, there holds $\operatorname{rank}(M) = \operatorname{rank}(\Sigma)$. (iii) According to Linear Algebra, the

factorization $M = VW^T$ implies

$$\operatorname{rank}(M) \leq \min\{\operatorname{rank}(V), \operatorname{rank}(W)\} \leq p.$$

To prove the converse implication, we may assume that $p \leq \min\{m, n\}$. Let $M = U\Sigma V^H$ be the singular value decomposition. We write $U = (U_0, U_1)$ and $V = (V_0, V_1)$ with $U_0 \in \mathbb{K}^{m\times p}$ and $V_0 \in \mathbb{K}^{n\times p}$. Moreover, let $S_p := \operatorname{diag}(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{p\times p}$. Since $\operatorname{rank}(M) \leq p$ implies $\sigma_{p+1} = \cdots = \sigma_{\min\{m,n\}} = 0$, there holds $M = U_0 S_p V_0^H$. Note that $U_0 S_p \in \mathbb{K}^{m\times p}$ so that we have obtained the claimed factorization. ■

3.1.2 Continuous Dependence of Singular Values

Next, we prove that the set

$$\mathbb{K}_p^{m\times n} := \big\{M \in \mathbb{K}^{m\times n} \,\big|\, \operatorname{rank}(M) \leq p\big\} \tag{3.1}$$

is a closed subset of $\mathbb{K}^{m\times n}$. To this end, we first prove that the singular values depend continuously on the matrix.

Proposition 3.3. *The singular values σ_j depend continuously on the matrix $M \in \mathbb{K}^{m\times n}$.*

Proof. The proof is mainly based on eigenvalue theory: A matrix $B \in \mathbb{K}^{n\times n}$ is self-adjoint if $B = B^H$. For any self-adjoint matrix $B \in \mathbb{K}^{n\times n}$, let $\lambda(B) \in \mathbb{R}^n$ denote the vector of eigenvalues, which is ordered by $\lambda_1 \geq \cdots \geq \lambda_n$. For self-adjoint matrices $B, B_n \in \mathbb{K}^{n\times n}$ holds

$$\|\lambda(B) - \lambda(B_n)\|_\infty \leq \|B - B_n\|_2,$$

c.f. Plato [Theorem 12.14]. For a convergent sequence $M_n \in \mathbb{K}^{m\times n}$ with limit $M \in \mathbb{K}^{m\times n}$, we define $B := M^H M$ and $B_n := M_n^H M_n$. Then,

$$\lim_{n\to\infty} B_n = B, \quad \text{whence} \quad \lim_{n\to\infty} \lambda(B_n) = \lambda(B).$$

Since the (extended) vector $\sigma(M) \in \mathbb{R}^n$ of singular values reads $\sigma(M) = (\lambda_1(B)^{1/2}, \ldots, \lambda_n(B)^{1/2})$, we obtain that $\lim_{n\to\infty} \sigma(M_n) = \sigma(M)$. ■

Corollary 3.4. *The set $\mathbb{K}_p^{m\times n} := \big\{M \in \mathbb{K}^{m\times n} \,\big|\, \operatorname{rank}(M) \leq p\big\}$ is a closed subset of $\mathbb{K}^{m\times n}$.*

Proof. Let $M_n \in \mathbb{K}^{m\times n}$ be a convergent sequence with $\operatorname{rank}(M_n) \leq p$ and limit $M \in \mathbb{K}^{m\times n}$. We have to show that $\operatorname{rank}(M) \leq p$. From the last proposition, we obtain $\lim_n \sigma(M_n) = \sigma(M)$. Since $\sigma_j(M_n) = 0$ for $j > p$, we see $\sigma_j(M) = 0$ for $j > p$, which concludes the proof. ■

3.1.3 Relation of SVD with Frobenius Norm and Euclidean Operator Norm

We have already proven the estimate $\|\cdot\|_2 \leq \|\cdot\|_F$. However, the next corollary gives an impression on the overestimation when we use $\|\cdot\|_F$ to estimate $\|\cdot\|_2$ from above. We stress that the main part of the proof verifies that the Frobenius norm is invariant under orthogonal matrices.

Corollary 3.5. *For any matrix $M \in \mathbb{K}^{m\times n}$ with singular values σ_j holds*

$$\|M\|_2 = \sigma_1 \quad \textit{and} \quad \|M\|_F = \Big(\sum_{j=1}^{\min\{m,n\}} \sigma_j^2 \Big)^{1/2}. \tag{3.2}$$

Proof. Let $M = U\Sigma V^H$ be the singular value decomposition. From the orthogonality of the matrices U and V^H, we obtain

$$\|M\|_2 = \|U\Sigma V^H\|_2 = \|\Sigma\|_2 = \sigma_1.$$

Next, we prove that the Frobenius norm is invariant under orthogonal matrices, i.e.

$$\|UB\|_F = \|B\|_F = \|BV^H\|_F \quad \text{for all } B \in \mathbb{K}^{m\times n}.$$

To that end, recall that the Frobenius norm stems from a scalar product:

$$\|UB\|_F^2 = \sum_{j,k} (UB)_{jk}\overline{(UB)_{jk}} = \sum_{j,k} \Big(\sum_i u_{ji} b_{ik} \Big) \Big(\sum_\ell \overline{u_{j\ell} b_{\ell k}} \Big) = \sum_{i,k,\ell} b_{ik} \overline{b_{\ell k}} \Big(\sum_j u_{ji} \overline{u_{j\ell}} \Big).$$

Note that the inner sum is the scalar product of the i-th and the ℓ-th column of U and thus equals $\delta_{i\ell}$ according to the orthogonality of U. Therefore,

$$\|UB\|_F^2 = \sum_{i,k,\ell} b_{ik}\overline{b_{\ell k}}\delta_{i\ell} = \sum_{k,\ell} |b_{\ell k}|^2 = \|B\|_F^2.$$

Analogously, one verifies $\|BV^H\|_F = \|B\|_F$ since V^H is an orthogonal matrix. Altogether, we thus obtain

$$\|M\|_F = \|U\Sigma V^H\|_F = \|\Sigma\|_F = \Big(\sum_{j=1}^{\min\{m,n\}} \sigma_j^2 \Big)^{1/2}$$

which concludes the proof. ■

Remark. As a consequence of Corollary 3.5, we see that the norm equivalence estimates

$$\|M\|_2 \le \|M\|_F \le \min\{\sqrt{m}, \sqrt{n}\}\, \|M\|_2 \quad \text{for all } M \in \mathbb{K}^{m\times n}$$

are sharp in the sense that there are matrices M for which the lower resp. upper bound is attained. □

3.1.4 Best Approximation by Low-Rank Matrices

Next, we want to introduce a best approximation operator

$$\mathcal{T}_p : \mathbb{K}^{m\times n} \to \mathbb{K}_p^{m\times n} := \big\{ M \in \mathbb{K}^{m\times n} \,\big|\, \operatorname{rank}(M) \le p \big\} \tag{3.3}$$

onto the rank-p-matrices with respect to the Frobenius and ℓ_2-operator norm. It turns out that both are given in terms of the singular value decomposition.

Theorem 3.6. *Let $M \in \mathbb{K}^{m\times n}$ with singular value decomposition $M = U\Sigma V^H$ and singular values σ_j and let $p \in \mathbb{N}$ with $p \leq \min\{m,n\}$. We write $U \in \mathbb{K}^{m\times m}$ and $V \in \mathbb{K}^{n\times n}$ in the form $U = (U_0, U_1)$ and $V = (V_0, V_1)$ with $U_0 \in \mathbb{K}^{m\times p}$ and $V_0 \in \mathbb{K}^{n\times p}$. Moreover, we define $S_p := \operatorname{diag}(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{p\times p}$. Then, the matrix*

$$\mathcal{T}_p M := U_0 S_p V_0^H \in \mathbb{K}_p^{m\times n} := \big\{ M \in \mathbb{K}^{m\times n} \,\big|\, \operatorname{rank}(M) \leq p \big\} \tag{3.4}$$

is a best approximation of M with respect to the Frobenius norm, i.e.

$$\|M - \mathcal{T}_p M\|_F = \min_{M_p \in \mathbb{K}_p^{m\times n}} \|M - M_p\|_F = \Big(\sum_{j=p+1}^{\operatorname{rank}(M)} \sigma_j^2 \Big)^{1/2}. \tag{3.5}$$

Furthermore, there holds

$$\|M - \mathcal{T}_p M\|_2 = \min_{M_p \in \mathbb{K}_p^{m\times n}} \|M - M_p\|_2 = \sigma_{p+1}. \tag{3.6}$$

Remark. One can show that the best approximation $\mathcal{T}_p M$ of a matrix $M \in \mathbb{K}^{m\times n}$ is in general not unique. The reason for this is pretty simple: Let $M = U\Sigma V^H$ be the singular value decomposition and let $u_j \in \mathbb{K}^m$ and $v_j \in \mathbb{K}^n$ denote the columns of U and V, respectively. For $x \in \mathbb{K}^n$, there holds

$$Mx = U\Sigma V^H x = \sum_{j=1}^{\operatorname{rank}(M)} \sigma_j (v_j \cdot x) u_j.$$

Provided $\sigma_p = \sigma_{p+1}$, we may interchange the p-th and $(p+1)$-st columns of U and V, respectively, without changing M. However, in general, there holds

$$(v_p \cdot x) u_p \neq (v_{p+1} \cdot x) u_{p+1}$$

so that we are led to two different best approximations M. □

Proof of Theorem 3.6. Without loss of generality, we may assume that $\operatorname{rank}(M) > p$. Otherwise the best approximation errors are zero and uniquely attained for $M_p = M$. According to the definition of $\mathcal{T}_p M$ and Corollary 3.5, there holds

$$\|M - \mathcal{T}_p M\|_2 = \sigma_{p+1} \quad \text{and} \quad \|M - \mathcal{T}_p M\|_F = \Big(\sum_{j>p} \sigma_j^2 \Big)^{1/2}.$$

It remains to prove that these bounds are optimal: Let $M_p \in \mathbb{K}_p^{m\times n}$. Let $u_j \in \mathbb{K}^m$ and $v_j \in \mathbb{K}^n$ denote the columns of $U \in \mathbb{K}^{m\times m}$ and $V \in \mathbb{K}^{n\times n}$, respectively.

1. step. By induction, we construct orthonormal vectors $z_{p+1}, \ldots, z_{\operatorname{rank}(M)} \in \mathbb{K}^n$ with

$$z_j \in \ker(M_p) \cap \operatorname{span}\{v_1, \ldots, v_j\} =: V_j \quad \text{for all } j = p+1, \ldots, \operatorname{rank}(M), \tag{3.7}$$

where $v_j \in \mathbb{K}^n$ are the columns of V. The induction starts with $j = p+1$. We use a dimension argument to show that $V_{p+1} \neq \{0\}$. Note that $\dim\ker(M_p) \geq n - p$. Hence,

$$\dim\ker(M_p) + j \geq (n-p) + (p+1) = n+1 \quad \text{and thus} \quad \dim V_{p+1} \geq 1.$$

This allows to choose a unit vector $z_{p+1} \in V_{p+1}$. In the induction step, we are given $z_{p+1}, \dots, z_j$ and have to prove the existence of z_{j+1}: Again note that

$$\dim \ker(M_p) + (j+1) \geq (n-p) + (j+1), \quad \text{whence} \quad \dim V_{j+1} \geq j - p + 1.$$

So far, we have $j - p$ orthonormal vectors $z_i \in V_i \subseteq V_{j+1}$ for $i = p+1, \dots, j$. Therefore, there is at least one vector $z_{j+1} \in V_{j+1}$ such that $\{z_{p+1}, \dots, z_{j+1}\}$ is an orthonormal system.

2. step. There holds $\|M - M_p\|_2 \geq \sigma_{p+1}$: Since the columns v_j of V are orthonormal, we have

$$Mz_{p+1} = U\Sigma V^H z_{p+1} = \sum_{j=1}^{\operatorname{rank}(M)} \sigma_j (v_j \cdot z_{p+1}) u_j = \sum_{j=1}^{p+1} \sigma_j (v_j \cdot z_{p+1}) u_j.$$

Moreover, $z_{p+1} \in \ker(M_p)$ and $\|z_{p+1}\|_2 = 1$ imply

$$\|M - M_p\|_2^2 \geq \|(M - M_p) z_{p+1}\|_2^2 = \|M z_{p+1}\|_2^2 = \sum_{j=1}^{p+1} \sigma_j^2 |v_j \cdot z_{p+1}|^2 \geq \sigma_{p+1}^2 \sum_{j=1}^{p+1} |v_j \cdot z_{p+1}|^2,$$

where we have also used that the columns u_j of U are orthonormal. From $\sum_{j=1}^{p+1} |v_j \cdot z_{p+1}|^2 = \|z_{p+1}\|_2^2 = 1$, we conclude the proof.

3. step. There holds $\|M - M_p\|_F^2 \geq \sum_{j=p+1}^{\operatorname{rank}(M)} \sigma_j^2$: The orthonormal system $\{z_{p+1}, \dots, z_{\operatorname{rank}(M)}\} \subset \mathbb{K}^n$ may be extended to an orthonormal basis $\{z_1, \dots, z_n\}$ of $\mathbb{K}^n$. We write this in form of the orthogonal matrix $Z \in \mathbb{K}^{n \times n}$ with columns z_j. From the invariance of the Frobenius norm under Z, we infer

$$\begin{aligned} \|M - M_p\|_F^2 = \|(M - M_p) Z\|_F^2 = \sum_{j=1}^{n} \|(M - M_p) z_j\|_2^2 &\geq \sum_{j=p+1}^{\operatorname{rank}(M)} \|(M - M_p) z_j\|_2^2 \\ &\geq \sum_{j=p+1}^{\operatorname{rank}(M)} \sigma_j^2, \end{aligned}$$

where the estimates $\|(M - M_p) z_j\|_2 \geq \sigma_j$ follow as in step 2. ■

3.1.5 Numerical Computation of Singular Value Decomposition

According to Golub, van Loan [p. 254, Section 5.4.5], the arithmetic complexity of the nowadays algorithms to compute the singular value decomposition of a matrix $M \in \mathbb{K}^{m \times n}$ is

$$\text{either} \quad 4m^2 n + 8mn^2 + 9n^3 \quad \text{or} \quad 4m^2 n + 22n^3 \tag{3.8}$$

in dependence on the chosen algorithm. In particular, for $m = n$, the first algorithm yields a complexity of $21m^3$ to compute the singular value decomposition. However, for rank-p-matrices, which are given in factorized form, the singular value decomposition can be computed much more efficiently. This observation (and concrete algorithm) will be the heart of the matter for the later $\mathcal{H}$-algorithms.

Lemma 3.7. *Let $M \in \mathbb{K}^{m\times n}$ be given in factorized form $M = VW^H$ with matrices $V \in \mathbb{K}^{m\times p}$ and $W \in \mathbb{K}^{n\times p}$. Then, the singular value decomposition can be computed in*

$$6p^2(m+n) + 23p^3 = \mathcal{O}\big(p^2(m+n) + p^3\big) \tag{3.9}$$

arithmetic operations. The operator norm $\|M\|_2$ is computed with the same complexity. The computation of the Frobenius norm $\|M\|_F$ needs additional $2p$ operations.

Proof. We write the proof in an algorithmic way so that it becomes clear what has to be implemented. Here, QR-factorization of a matrix $M \in \mathbb{K}^{m\times n}$ means the reduced form $M = QR$ with $R \in \mathbb{K}^{n\times n}$ a right upper triangular matrix and $Q \in \mathbb{K}^{m\times n}$ a matrix with orthogonal columns. The computational complexity for the QR-factorization is taken from GOLUB, VAN LOAN [p. 232, Section 5.2.9]. We consider the factorization $M = VW^H$ and proceed as follows:

- Compute the QR-factorization $V = Q_V R_V$ in $4mp^2$ operations.
- Compute the QR-factorization $W = Q_W R_W$ in $4np^2$ operations.
- Compute matrix-matrix multiplication $R := R_V R_W^H$ in (less than) $2p^3$ operations.
- Compute singular value decomposition $R = U_0 \Sigma V_0^H$ in $21p^3$ operations.
- Compute orthogonal matrix $U_1 := Q_V U_0$ in (less than) $2mp^2$ operations.
- Compute orthogonal matrix $V_1 := Q_W V_0$ in (less than) $2np^2$ operations.

Then, we have used $6p^2(m+n) + 23p^3$ arithmetic operations to compute the factorization

$$M = VW^H = Q_V R_V R_W^H Q_W^H = Q_V (U_0 \Sigma V_0^H) Q_W^H = (Q_V U_0)\Sigma (Q_W V_0)^H = U_1 \Sigma V_1^H.$$

Note that this is the singular value decomposition of M. From Corollary 3.5, we derive that $\|M\|_2 = \sigma_1$ is obtained as a side result. Moreover, $\|M\|_F = \big(\sum_{j=1}^{p} \sigma_j^2\big)^{1/2}$ needs additional $2p-1$ arithmetic operations plus the computation of the square root. ■

Corollary 3.8. *Assume that $M \in \mathbb{K}^{m\times n}$ is given in factorized form $M = VW^H$ with $V \in \mathbb{K}^{m\times p}$ and $W \in \mathbb{K}^{n\times p}$. Then, for $p' \le p$, it needs*

$$6p^2(m+n) + 23p^3 + mp' = \mathcal{O}\big(p^2(m+n) + p^3\big) \tag{3.10}$$

operations to compute matrices $V_{p'} \in \mathbb{K}^{m\times p'}$ and $W_{p'} \in \mathbb{K}^{n\times p'}$ such that

$$\mathcal{T}_{p'} M = V_{p'} W_{p'}^H. \tag{3.11}$$

Proof. We apply Lemma 3.7 to compute the reduced singular value decomposition $M = U_p S_p W_p^H$ in $6p^2(m+n) + 23p^3$ operations. With $U_{p'}$, $S_{p'}$, and $W_{p'}$ the first p' columns of U_p, S_p, and W_p, respectively, it only remains to compute the matrix-matrix product $V_{p'} := U_{p'} S_{p'} \in \mathbb{K}^{m\times p'}$. Since $S_{p'}$ is a diagonal matrix, this is done in mp' operations. ■

3.2 $\mathcal{H}$-Matrices Revisited (25.05.2009)

In this section, we take a second look at hierarchical matrices. Throughout and in the subsequent sections, we use the following assumptions:

- Let $\mathbb{T}_{\mathcal{I}}$ and $\mathbb{T}_{\mathcal{J}}$ be cluster trees for index sets $\mathcal{I} := \{1, \dots, m\}$ and $\mathcal{J} := \{1, \dots, n\}$, respectively.
- Let $\mathbb{A}$ be a given admissibility condition, and let the block partitioning $\mathbb{P}$ built by Algorithm 2.3.
- Let $p \in \mathbb{N}$ be the local rank of the $\mathcal{H}$-matrix.

First, we now formalize Algorithm 2.3, which creates the block partitioning $\mathbb{P}$: A second look at this algorithm reveals that we are building (but not storing) a quad tree $\mathbb{B}$, i.e. each node σ satisfies either $\operatorname{sons}(\sigma) = \emptyset$ or $|\operatorname{sons}(\sigma)| = 4$. The block partitioning $\mathbb{P}$ is nothing but the set of leaves of $\mathbb{B}$, which are classified into admissible and inadmissible leaves.

3.2.1 Hierarchical Trees

Definition. A tree $\mathbb{B}$ is a **hierarchical tree** (or **$\mathcal{H}$-tree**) with respect to an index set $\mathcal{I}$, provided $\operatorname{root}(\mathbb{B}) = \mathcal{I}$ and provided all nodes $\sigma \in \mathbb{B}$ satisfy the following properties:

- $\sigma \subseteq \mathcal{I}$ and $\sigma \neq \emptyset$,
- σ is either a leaf, i.e. $\operatorname{sons}(\sigma) = \emptyset$, or $|\operatorname{sons}(\sigma)| \geq 2$,
- if σ is not a leaf, it is the disjoint union of its sons, i.e. $\sigma = \cup \operatorname{sons}(\sigma)$ and $\sigma' \cap \sigma'' = \emptyset$ for all $\sigma', \sigma'' \in \operatorname{sons}(\sigma)$.

For an $\mathcal{H}$-tree $\mathbb{B}$, we define the **level function** $\operatorname{level} : \mathbb{B} \to \mathbb{N}_0$ inductively by

$$\operatorname{level}(\mathcal{I}) = 0 \quad \text{and} \quad \operatorname{level}(\sigma') = \operatorname{level}(\sigma) + 1 \quad \text{for all } \sigma' \in \operatorname{sons}(\sigma).$$

Finally, we define the **depth** by $\operatorname{depth}(\mathbb{B}) := \max_{\sigma \in \mathbb{B}} \operatorname{level}(\sigma)$.

A first elementary but important observation is that the leaves $\mathbb{P}$ of $\mathbb{B}$ yield a partitioning of the index set $\mathcal{I}$ and that the number of nodes in $\mathbb{B}$ is of order $\mathcal{O}(n)$ with $n = |\mathcal{I}|$.

Lemma 3.9. *Let $\mathbb{B}$ be an $\mathcal{H}$-tree for the index set $\mathcal{I}$ with*

$$n := |\mathcal{I}| \quad \text{and} \quad m := \max\big\{2, \max_{\sigma \in \mathbb{B}} |\operatorname{sons}(\sigma)|\big\}. \tag{3.12}$$

Let $\mathbb{P}$ denote the set of leaves of $\mathbb{B}$. Then, the number of nodes in the tree is bounded by

$$|\mathbb{B}| \leq m|\mathbb{P}| - 1. \tag{3.13}$$

Moreover, $\mathbb{P}$ is a partitioning of $\mathcal{I}$.

***Proof.* 1. step.** $|\mathbb{B}| \leq m|\mathbb{P}| - 1$ is proven by induction on n: For $n = 1$, there holds $|\mathbb{B}| = 1 = |\mathbb{P}|$ and $m = 2$. For $n > 1$, we consider $\sigma' \in \operatorname{sons}(\mathcal{I})$. Then, $|\sigma'| \leq n - 1$ and the $\mathcal{H}$-subtree $\mathbb{B}'$ starting from root σ' satisfies the induction hypothesis, which yields $|\mathbb{B}'| \leq m|\mathbb{P}'| - 1$. Summing

up all the sons, we are led to

$$|\mathbb{B}| \leq 1 + (m|\mathbb{P}| - |\mathrm{sons}(\mathcal{I})|) \leq m|\mathbb{P}| - 1$$

since each leaf of $\mathbb{B}$ is a leaf in precisely one subtree.

2. step. To prove that $\mathbb{P}$ is a partitioning of $\mathcal{I}$, one may proceed inductively. For any $x \in \mathcal{I}$, there are unique nodes $\sigma_0, \ldots, \sigma_\ell \in \mathbb{B}$ such that

$$x \in \sigma_{j+1} \in \mathrm{sons}(\sigma_j) \quad \text{and} \quad \mathrm{sons}(\sigma_\ell) = \emptyset.$$

In particular, this proves $x \in \sigma_\ell \in \mathbb{P}$. Therefore, $\mathbb{P}$ is a covering of $\mathcal{I}$. Finally, it remains to prove that $\sigma \cap \tau = \emptyset$ for $\sigma, \tau \in \mathbb{P}$ with $\sigma \neq \tau$. Inductively, we obtain unique nodes $\sigma_0, \ldots, \sigma_k$ and $\tau_0, \ldots, \tau_\ell$ with

$$\sigma = \sigma_k \quad \text{and} \quad \sigma_{j+1} \in \mathrm{sons}(\sigma_j) \qquad \text{as well as} \qquad \tau = \tau_\ell \quad \text{and} \quad \tau_{j+1} \in \mathrm{sons}(\tau_j).$$

We choose the maximal index i with $\sigma_i = \tau_i$. Clearly, $i < \min\{k, \ell\}$ since $\sigma = \sigma_k$ and $\tau = \tau_\ell$ are leaves. Consequently, $\sigma_{i+1} \neq \tau_{i+1}$ are sons of $\sigma_i = \tau_i$. Therefore, $\emptyset = \sigma_{i+1} \cap \tau_{i+1} \supseteq \sigma \cap \tau$. ■

Obviously, a cluster tree is nothing but an $\mathcal{H}$-tree. Moreover, two cluster trees $\mathbb{T}_\mathcal{I}$ and $\mathbb{T}_\mathcal{J}$ for index sets $\mathcal{I}$ and $\mathcal{J}$, respectively, induce an $\mathcal{H}$-tree for the index set $\mathcal{I} \times \mathcal{J}$. The following definition is nothing but a mathematical formalism of Algorithm 2.3.

Definition. Given $\mathcal{H}$-trees $\mathbb{T}_\mathcal{I}$ and $\mathbb{T}_\mathcal{J}$ for index sets $\mathcal{I}$ and $\mathcal{J}$, respectively, we define the **induced block tree** $\mathbb{B} := \mathbb{B}(\mathbb{T}_\mathcal{I}, \mathbb{T}_\mathcal{J})$ inductively as follows:

- $\mathrm{root}(\mathbb{B}) := \mathcal{I} \times \mathcal{J}$,
- $\mathrm{sons}(\sigma \times \tau) := \big\{\sigma' \times \tau' \,\big|\, \sigma' \in \mathrm{sons}(\sigma), \tau' \in \mathrm{sons}(\tau)\big\}$.

Moreover, if $\mathbb{A}$ is an admissibility condition, we define $\mathbb{B} := \mathbb{B}(\mathbb{T}_\mathcal{I}, \mathbb{T}_\mathcal{J}, \mathbb{A})$ by the reduced son relation

- $\mathrm{sons}(\sigma \times \tau) := \begin{cases} \emptyset, & \sigma \times \tau \text{ is admissible,} \\ \big\{\sigma' \times \tau' \,\big|\, \sigma' \in \mathrm{sons}(\sigma), \tau' \in \mathrm{sons}(\tau)\big\}, & \text{else.} \end{cases}$

The proof of the following simple observation is left to the reader.

Lemma 3.10. *The block trees $\mathbb{B} = \mathbb{B}(\mathbb{T}_\mathcal{I}, \mathbb{T}_\mathcal{J})$ resp. $\mathbb{B} = \mathbb{B}(\mathbb{T}_\mathcal{I}, \mathbb{T}_\mathcal{J}, \mathbb{A})$ are $\mathcal{H}$-trees for $\mathcal{I} \times \mathcal{J}$ with* $\mathrm{depth}(\mathbb{B}) \leq \min\{\mathrm{depth}(\mathbb{T}_\mathcal{I}), \mathrm{depth}(\mathbb{T}_\mathcal{J})\}$. ■

We note that the tree $\mathbb{B} := \mathbb{B}(\mathbb{T}_\mathcal{I}, \mathbb{T}_\mathcal{J}, \mathbb{A})$ is the tree that is implicitly built by Algorithm 2.3, i.e., the leaves of $\mathbb{B}$ form the blockpartitioning $\mathbb{P}$ we were working with in Chapter 2.We now redefine the sparsity constant:

Definition. For an induced block tree $\mathbb{B} := \mathbb{B}(\mathbb{T}_\mathcal{I}, \mathbb{T}_\mathcal{J}, \mathbb{A})$ with set of leaves $\mathbb{P}$, we define

$$C_{\mathrm{sparse}}(\mathbb{P}) := \max\Big\{ \max_{\sigma \in \mathbb{T}_\mathcal{I}} \big|\big\{\tau \in \mathbb{T}_\mathcal{J} \,\big|\, (\sigma, \tau) \in \mathbb{P}\big\}\big|, \max_{\tau \in \mathbb{T}_\mathcal{J}} \big|\big\{\sigma \in \mathbb{T}_\mathcal{I} \,\big|\, (\sigma, \tau) \in \mathbb{P}\big\}\big| \Big\}, \tag{3.14}$$

$$C_{\mathrm{sparse}}(\mathbb{B}) := \max\Big\{ \max_{\sigma \in \mathbb{T}_\mathcal{I}} \big|\big\{\tau \in \mathbb{T}_\mathcal{J} \,\big|\, (\sigma, \tau) \in \mathbb{B}\big\}\big|, \max_{\tau \in \mathbb{T}_\mathcal{J}} \big|\big\{\sigma \in \mathbb{T}_\mathcal{I} \,\big|\, (\sigma, \tau) \in \mathbb{B}\big\}\big| \Big\}. \tag{3.15}$$

If the block tree $\mathbb{B}$ is clear from the context, we shall abbreviate $C_{\mathrm{sparse}} := C_{\mathrm{sparse}}(\mathbb{B})$.

Throughout, Chapter 2, we were writing C_{sparse} to abbreviate $C_{\text{sparse}}(\mathbb{P})$. From now on, we shall use C_{sparse} to abbreviate $C_{\text{sparse}}(\mathbb{B})$. Note that, by definition, $C_{\text{sparse}}(\mathbb{P}) \leq C_{\text{sparse}}(\mathbb{B})$. Therefore, the proven results even hold with the redefined sparsity constant.

3.2.2 Hierarchical Matrices

With Corollary 3.2 (iii), we can recall the definition of a hierarchical matrix:

Definition. Let $\mathbb{B} = \mathbb{B}_{\mathcal{I}\times\mathcal{J}}$ be a hierarchical tree with respect to $\mathcal{I}\times\mathcal{J}$. We assume that the leaves $\mathbb{P}$ of $\mathbb{B}$ are split into far field leaves $\mathbb{P}_{\text{far}}$ and near field leaves $\mathbb{P}_{\text{near}}$. We then define the set

$$\mathcal{H}(\mathbb{B},p) := \big\{A\in\mathbb{K}^{m\times n} \,\big|\, \forall(\sigma,\tau)\in\mathbb{P}_{\text{far}} \quad \operatorname{rank}(A|_{\sigma\times\tau}) \leq p\big\}. \tag{3.16}$$

We call a matrix $A\in\mathbb{K}^{m\times n}$ **hierarchical matrix**, if each admissible block $A|_{\sigma\times\tau}$ is given and stored in factorized form, i.e. there are matrices $V_{\sigma\tau}\in\mathbb{K}^{|\sigma|\times p}$ and $W_{\sigma\tau}\in\mathbb{K}^{|\tau|\times p}$ such that

$$A|_{\sigma\times\tau} = V_{\sigma\tau}W_{\sigma\tau}^H \quad \text{for all } (\sigma,\tau)\in\mathbb{P}_{\text{far}}. \tag{3.17}$$

If the block partitioning $\mathbb{B}$ is clear from the context, we shall abbreviate $\mathcal{H}(p) := \mathcal{H}(\mathbb{B},p)$.

Proposition 3.11. *The set $\mathcal{H}(\mathbb{B},p)$ of hierarchical matrices is a closed subset of $\mathbb{K}^{m\times n}$.*

Proof. Let $A_n\in\mathcal{H}(\mathbb{B},p)$ be a convergent sequence with limit $A\in\mathbb{K}^{m\times n}$. We only have to show that $\operatorname{rank}(A|_{\sigma\times\tau})\leq p$ for all $(\sigma,\tau)\in\mathbb{P}_{\text{far}}$. This follows as for Corollary 3.4. ■

3.2.3 Bestapproximation by $\mathcal{H}$-Matrices

In this section, we want to define an operator

$$\mathcal{T}_{\mathcal{H}(p)} : \mathbb{K}^{m\times n} \to \mathcal{H}(p) \tag{3.18}$$

which maps a matrix $A\in\mathbb{K}^{m\times n}$ to a best approximation in $\mathcal{H}(p)$ with respect to the Frobenius norm, i.e.

$$\|A - \mathcal{T}_{\mathcal{H}(p)}A\|_F = \min_{A_{\mathcal{H}}\in\mathcal{H}(p)} \|A - A_{\mathcal{H}}\|_F. \tag{3.19}$$

To that end, let $\mathcal{T}_p$ denote the truncation operator onto rank-p-matrices from Theorem 3.6. We may apply $\mathcal{T}_p$ to the far field blocks of an $\mathcal{H}$-matrix $A\in\mathcal{H}(p)$. This induces an operator $\mathcal{T}_{\mathcal{H}(p)} : \mathbb{K}^{m\times n}\to\mathcal{H}(p)$. This operator is formally defined by

$$(\mathcal{T}_{\mathcal{H}(p)}A)|_{\sigma\times\tau} := A|_{\sigma\times\tau} \quad \text{for } (\sigma,\tau)\in\mathbb{P}_{\text{near}} \tag{3.20}$$

and

$$(\mathcal{T}_{\mathcal{H}(p)}A)|_{\sigma\times\tau} := \mathcal{T}_p(A|_{\sigma\times\tau}) \quad \text{for } (\sigma,\tau)\in\mathbb{P}_{\text{far}}. \tag{3.21}$$

We will need the following corollary later-on for theoretical purposes only.

Corollary 3.12. *The operator $\mathcal{T}_{\mathcal{H}(p)}$ is a best approximation operator with respect to the Frobenius norm, i.e.*

$$\|A - \mathcal{T}_{\mathcal{H}(p)}A\|_F = \min_{A_{\mathcal{H}} \in \mathcal{H}(p)} \|A - A_{\mathcal{H}}\|_F \quad \text{for all } A \in \mathbb{K}^{m\times n}.$$

Proof. The bestapproximation $A_{\mathcal{H}} \in \mathcal{H}(p)$ of A is given if we choose the $\mathbb{P}$-blockwise bestapproximation for any block $A|_{\sigma\times\tau}$ with $(\sigma,\tau) \in \mathbb{P}$. This is obviously given by (3.20) in the near field. Moreover, Theorem 3.6 shows that (3.21) is the optimal choice for each far field block. ■

In theory, Corollary 3.12 allows the computation of the bestapproximation $\mathcal{T}_{\mathcal{H}(p)}A \in \mathcal{H}(p)$ of a dense matrix $A \in \mathbb{K}^{m\times n}$. However, in practice one aims to avoid to assemble the dense matrix $A \in \mathbb{K}^{m\times n}$ because of the quadratic complexity mn for assembly and storage. However, it is cheap to compute $\mathcal{T}_{\mathcal{H}(p')}A$ for an $\mathcal{H}$-matrix $A \in \mathcal{H}(p)$ and $p' < p$.

Proposition 3.13. *Given a hierarchical matrix $A \in \mathcal{H}(p)$ and $p' < p$, it needs*

$$23p^3|\mathbb{P}_{\text{far}}| + C_{\text{sparse}}(\text{depth}(\mathbb{P}) + 1)(6p^2 + p')(m + n) = \mathcal{O}(p^3N + p^2N \log N) \tag{3.22}$$

operations to compute $\mathcal{T}_{\mathcal{H}(p')}A$, where the admissible blocks are stored in factorized form and where $N = m + n$.

Proof. We only have to deal with the far field: According to Corollary 3.8, the computational complexity to compute $\mathcal{T}_{\mathcal{H}(p')}A$ is

$$\sum_{(\sigma,\tau)\in\mathbb{P}_{\text{far}}} \left(6p^2(|\sigma| + |\tau|) + 23p^3 + |\sigma|p'\right) \leq 23p^3|\mathbb{P}_{\text{far}}| + (6p^2 + p') \sum_{(\sigma,\tau)\in\mathbb{P}_{\text{far}}} (|\sigma| + |\tau|).$$

Above, we have already verified that

$$\sum_{(\sigma,\tau)\in\mathbb{P}_{\text{far}}} (|\sigma| + |\tau|) \leq \sum_{(\sigma,\tau)\in\mathbb{P}} (|\sigma| + |\tau|) \leq C_{\text{sparse}}(\text{depth}(\mathbb{P}) + 1)(m + n).$$

This concludes the proof. ■

3.3 Fast Computation of Norms (19.05.2009)

In this section we want to prove that the Frobenius norm of an $\mathcal{H}$-matrix A can be computed in almost linear complexity, which is another application of the singular value decomposition.

Frobenius Norm. The Frobenius norm of a near field block can be computed just by definition. For a far field block, recall that the singular value decomposition can be computed by Lemma 3.7 efficiently. Therefore, the combination with Lemma 3.5 yields an efficient algorithm for the computation of the Frobenius norm of a far field block.

Proposition 3.14. *For $A \in \mathcal{H}(p)$, the computation of the Frobenius norm $\|A\|_F$ can be done in*

$$\begin{aligned}&(23p^3 + 2p)|\mathbb{P}| + 2C_{\text{sparse}}(\text{depth}(\mathbb{P}) + 1)\max\{C_{\text{leaf}}, 3p^2\}(m+n)\\ &\quad = \mathcal{O}(p^3 N + p^2 N \log N)\end{aligned} \tag{3.23}$$

operations, where $N = m + n$.

Proof. For a near field block $(\sigma, \tau) \in \mathbb{P}_{\text{near}}$, the computation of $\|A|_{\sigma\times\tau}\|_F^2$ is done directly and needs $2|\sigma||\tau| - 1$ arithmetic operations. Note that $|\sigma||\tau| \leq C_{\text{leaf}}(|\sigma| + |\tau|)$ according to Algorithm 2.3. For a far field block $(\sigma, \tau) \in \mathbb{P}_{\text{far}}$, the computation of $\|A|_{\sigma\times\tau}\|_F^2$ is done according to Lemma 3.7 in $6p^2(|\sigma| + |\tau|) + 23p^3 + 2p - 1$ operations. With

$$\|A\|_F^2 = \sum_{(\sigma,\tau)\in\mathbb{P}} \|A|_{\sigma\times\tau}\|_F^2,$$

the computational complexity can be estimated by

$$\begin{aligned}&\sum_{(\sigma,\tau)\in\mathbb{P}_{\text{near}}} \big(2|\sigma||\tau| - 1\big) + \sum_{(\sigma,\tau)\in\mathbb{P}_{\text{far}}} \big(6p^2(|\sigma| + |\tau|) + 23p^3 + 2p - 1\big)\\ &\qquad \leq -|\mathbb{P}_{\text{near}}| + (23p^3 + 2p - 1)|\mathbb{P}_{\text{far}}| + \max\{2C_{\text{leaf}}, 6p^2\} \sum_{(\sigma,\tau)\in\mathbb{P}} (|\sigma| + |\tau|)\end{aligned}$$

According to elementary calculations, cf. Equation (2.9) on page 7, we have

$$\sum_{(\sigma,\tau)\in\mathbb{P}} (|\sigma| + |\tau|) \leq C_{\text{sparse}}(\text{depth}(\mathbb{P}) + 1)(m + n).$$

Altogether, we are thus led to a computational complexity less than

$$(23p^3 + 2p)|\mathbb{P}| + 2C_{\text{sparse}}(\text{depth}(\mathbb{P}) + 1)\max\{C_{\text{leaf}}, 3p^2\}(m + n)$$

to compute $\|A\|_F$. ■

ℓ_2-Operator Norm. The ℓ_2-operator norm is harder to compute than the Frobenius norm. In case that $A \in \mathcal{H}(p)$ is positive definite and self-adjoint, i.e. $A = A^H \in \mathbb{K}^{n\times n}$, the singular values σ_j of A are just the (real and positive) eigenvalues. Therefore, one can compute the largest eigenvalue σ_1 of A by **power iteration**, which is stated in the following:

- Let $x_0 \in \mathbb{K}^n$ be an arbitrary starting vector
- Define $x_k := Ax_{k-1}$ and the Rayleigh quotient $r_k := \frac{x_k \cdot x_{k-1}}{\|x_{k-1}\|_2^2}$ for $k \in \mathbb{N}$.

Note that each step of the power iteration only needs a (fast) matrix-vector multiplication with the $\mathcal{H}$-matrix A as well as some additional operations with linear complexity, e.g. computing the scalar product or the ℓ_2-vector norm. Under some weak assumptions on x_0, the sequence r_k of the Rayleigh quotients converges pretty fast to $\sigma_1 = \|A\|_2$. For details the reader is, e.g., referred to Plato [p. 310, Theorem 13.36].

The following lemma gives a lower and upper bound for the ℓ_2-operator norm and the reader may prove as an exercise that both bounds are in fact norms on $\mathbb{K}^{m\times n}$. We stress that the

lower as well as the upper bound can be computed efficiently for hierarchical matrices. The detailed complexity analysis is left to the reader but follows easily along the lines of the proof of Proposition 3.14.

Lemma 3.15. *For any matrix $A \in \mathbb{K}^{m\times n}$, there holds*

$$\max_{(\sigma,\tau)\in\mathbb{P}} \|A|_{\sigma\times\tau}\|_2 \le \|A\|_2 \le C_{\mathrm{sparse}} \sum_{\ell=0}^{\mathrm{depth}(\mathbb{P})} \max_{\substack{(\sigma,\tau)\in\mathbb{P}\\ \mathrm{level}(\sigma)=\ell}} \|A|_{\sigma\times\tau}\|_2 \tag{3.24}$$

Proof. The proof of the lower bound is only notationally challenging. We use a sloppy notation and simply extend vectors $x \in \mathbb{K}^{|\tau|} \equiv \mathbb{K}^\tau$ to vectors $\widehat{x} \in \mathbb{K}^n$ by

$$\widehat{x}_j := \begin{cases} x_j & \text{for } j \in \tau, \\ 0 & \text{else.} \end{cases}$$

Let $(\sigma,\tau) \in \mathbb{P}$ be an arbitrary block. Then,

$$\|A|_{\sigma\times\tau} x\|_2 \le \|A|_{\mathcal{I}\times\tau} x\|_2 = \|A\widehat{x}\|_2 \le \|A\|_2\|\widehat{x}\|_2 = \|A\|_2\|x\|_2,$$

which yields $\|A|_{\sigma\times\tau}\|_2 \le \|A\|_2$. For the proof of the upper bound in (3.24), we define

$$\varepsilon_\ell := \max_{\substack{(\sigma,\tau)\in\mathbb{P}\\ \mathrm{level}(\sigma)=\ell}} \|A|_{\sigma\times\tau}\|_2$$

to abbreviate notation. For $x \in \mathbb{K}^n$ and $y := Ax \in \mathbb{K}^m$, we have

$$\|Ax\|_2^2 = y\cdot(Ax) = \sum_j y_j (Ax)_j = \sum_{j,k} y_j A_{jk} x_k = \sum_{(\sigma,\tau)\in\mathbb{P}} y|_\sigma \cdot (A|_{\sigma\times\tau} x|_\tau)$$

since $\mathbb{P}$ is a partitioning of the summation domain $\mathcal{I}\times\mathcal{J}$. Therefore,

$$\begin{aligned}\|Ax\|_2^2 \le \sum_{(\sigma,\tau)\in\mathbb{P}} \|y|_\sigma\|_2 \|A|_{\sigma\times\tau}\|_2 \|x|_\tau\|_2 &\le \sum_{(\sigma,\tau)\in\mathbb{P}} (\varepsilon_{\mathrm{level}(\sigma)}^{1/2}\|y|_\sigma\|_2)(\varepsilon_{\mathrm{level}(\sigma)}^{1/2}\|x|_\tau\|_2) \\ &\le \Big(\sum_{(\sigma,\tau)\in\mathbb{P}} \varepsilon_{\mathrm{level}(\sigma)}\|y|_\sigma\|_2^2\Big)^{1/2}\Big(\sum_{(\sigma,\tau)\in\mathbb{P}} \varepsilon_{\mathrm{level}(\sigma)}\|x|_\tau\|_2^2\Big)^{1/2}\end{aligned}$$

Since $\mathrm{level}(\sigma) = \mathrm{level}(\tau)$ for $(\sigma,\tau)\in\mathbb{P}$, either sum can be estimated as follows:

$$\begin{aligned}\sum_{(\sigma,\tau)\in\mathbb{P}} \varepsilon_{\mathrm{level}(\sigma)}\|y|_\sigma\|_2^2 = \sum_{\ell=0}^{\mathrm{depth}(\mathbb{P})} \sum_{\substack{\sigma\in\mathbb{T}_\mathcal{I}\\ \mathrm{level}(\sigma)=\ell}} \sum_{\substack{\tau\in\mathbb{T}_\mathcal{J}\\ (\sigma,\tau)\in\mathbb{P}}} \varepsilon_\ell\|y|_\sigma\|_2^2 &\le C_{\mathrm{sparse}} \sum_{\ell=0}^{\mathrm{depth}(\mathbb{P})} \varepsilon_\ell \sum_{\substack{\sigma\in\mathbb{T}_\mathcal{I}\\ \mathrm{level}(\sigma)=\ell}} \|y|_\sigma\|_2^2 \\ &\le C_{\mathrm{sparse}}\|y\|_2^2 \sum_{\ell=0}^{\mathrm{depth}(\mathbb{P})} \varepsilon_\ell.\end{aligned}$$

This yields

$$\|Ax\|_2^2 \le \Big(C_{\mathrm{sparse}}\|y\|_2^2 \sum_{\ell=0}^{\mathrm{depth}(\mathbb{P})} \varepsilon_\ell\Big)^{1/2}\Big(C_{\mathrm{sparse}}\|x\|_2^2 \sum_{\ell=0}^{\mathrm{depth}(\mathbb{P})} \varepsilon_\ell\Big)^{1/2} = C_{\mathrm{sparse}}\|x\|_2\|y\|_2 \sum_{\ell=0}^{\mathrm{depth}(\mathbb{P})} \varepsilon_\ell$$

and thus $\|Ax\|_2 \le \|x\|_2$ according to $y = Ax$. Taking the supremum over all $x \in \mathbb{K}^n$, we conclude the proof. ∎

3.4 Re-Compression of $\mathcal{H}$-Matrices (25.05.2009)

Normally, an $\mathcal{H}$-matrix $A \in \mathcal{H}(p)$ is assembled using a priori knowledge: For instance, if we build the $\mathcal{H}$-matrix by interpolation, the cluster tree and the admissibility condition as well as the degree of the interpolation operator have to be chosen a priori. The analysis of the preceding chapter is a typical a priori analysis which states asymptotics and convergence for a broad class of examples —including the so-called worst case. Therefore, the developed algorithms may be too pessimistic in practice. Although the asymptotic complexity is nearly optimal, this leads to large numerical constants which are hidden in the big-$\mathcal{O}(\cdot)$-notation.

In this section, the strategy is therefore the following: We assume that we have assembled an $\mathcal{H}$-matrix, e.g., by interpolation. The aim now is to drop further data without loosing accuracy of the approximation. Given $A \in \mathcal{H}(\mathbb{T}_\mathcal{I}, \mathbb{T}_\mathcal{J}, \mathbb{A}, p)$ and $\varepsilon_{\text{tol}} > 0$, we seek for another$\mathcal{H}$-matrix $A^{\text{comp}} \in \mathcal{H}(\mathbb{T}_\mathcal{I}, \mathbb{T}_\mathcal{J}, \mathbb{A}^{\text{comp}}, p^{\text{comp}})$ with the following properties:

- A^{comp} needs less storage requirements and allows for faster computations.
- The admissibility condition $\mathbb{A}^{\text{comp}}$ is weaker than $\mathbb{A}$, i.e.

  $$\forall (\sigma, \tau) \in \mathbb{T}_\mathcal{I} \times \mathbb{T}_\mathcal{J} \quad \big[(\sigma, \tau) \text{ is } \mathbb{A}\text{-admissible} \Rightarrow (\sigma, \tau) \text{ is } \mathbb{A}^{\text{comp}}\text{-admissible}\big],$$

  and thus leads to more and probably larger admissible blocks in $A^{\text{comp}} \in \mathcal{H}(\mathbb{A}^{\text{comp}}, p^{\text{comp}})$ when compared to $A \in \mathcal{H}(\mathbb{A}, p)$.
- p^{comp} is chosen for each admissible block individually and as small as possible.
- We can control the relative error

  $$\frac{\|A - A^{\text{comp}}\|}{\|A\|} \leq \varepsilon_{\text{tol}}.$$

The re-compression strategy is split into three parts:

- Re-compression of admissible blocks: The rank of an admissible block $A|_{\sigma\times\tau} = VW^H$ with $V \in \mathbb{K}^{|\sigma|\times p}$ and $W \in \mathbb{K}^{|\tau|\times p}$ may satisfy $p' := \operatorname{rank}(A|_{\sigma\times\tau}) \ll p$. Therefore, it may be advantageous to build thinner matrices $V_{p'} \in \mathbb{K}^{|\sigma|\times p'}$ and $W_{p'} \in \mathbb{K}^{|\tau|\times p'}$ such that $A|_{\sigma\times\tau} = V_{p'} W_{p'}^H$.
- Re-compression of inadmissible blocks: In general, the admissibility condition is a sufficient criterion to decide whether a block is admissible or not. However, this leads in general to more inadmissible blocks than necessary. Because of the cheaper handling of admissible blocks, we should check whether an inadmissible block can be converted to an admissible block!
- Re-compression of block partitioning: The block partitioning $\mathbb{P}$ may consist of too many blocks. In particular, the depth of the block partitioning may be larger than necessary.

We stress that all compression steps are usually done on-the-fly while the assembly of the matrix.

3.4.1 Blockwise Recompression

The following lemma explains the compression strategy.

Lemma 3.16. *Let $A \in \mathcal{H}(p)$ and $\varepsilon > 0$. We define another matrix $\widetilde{A} \in \mathcal{H}(p)$ as follows: For each block $(\sigma,\tau) \in \mathbb{P}$ and $r := \operatorname{rank}(A|_{\sigma\times\tau})$, we seek $p' \in \{1,...,r-1\}$ with*

$$\sigma_{p'+1}(A|_{\sigma\times\tau}) \le \varepsilon\, \sigma_1(A|_{\sigma\times\tau}), \tag{3.25}$$

and choose $p' := r$ provided p' does not exist. We then define

$$\widetilde{A}|_{\sigma\times\tau} := \mathcal{T}_{p'}(A|_{\sigma\times\tau}). \tag{3.26}$$

The relative error between A and $\widetilde{A}$ then satisfies

$$\frac{\|A-\widetilde{A}\|_2}{\|A\|_2} \le C_{\mathrm{sparse}}(\operatorname{depth}(\mathbb{P})+1)\,\varepsilon \tag{3.27}$$

Proof. Note the the chosen $p' = p'(\sigma,\tau)$ may vary for each block $(\sigma,\tau) \in \mathbb{P}$. For notational simplicity, we skip the dependence of (σ,τ) in the following and define $\sigma_{r+1}(A|_{\sigma\times\tau}) := 0$ for $r := \operatorname{rank}(A|_{\sigma\times\tau})$. The error estimate in Lemma 3.15 implies

$$\begin{aligned}
\|A-\widetilde{A}\|_2 &\le C_{\mathrm{sparse}} \sum_{\ell=0}^{\operatorname{depth}(\mathbb{P})} \max_{\substack{(\sigma,\tau)\in\mathbb{P}\\ \operatorname{level}(\sigma)=\ell}} \|A|_{\sigma\times\tau} - \widetilde{A}|_{\sigma\times\tau}\|_2 \\
&= C_{\mathrm{sparse}} \sum_{\ell=0}^{\operatorname{depth}(\mathbb{P})} \max_{\substack{(\sigma,\tau)\in\mathbb{P}_{\mathrm{far}}\\ \operatorname{level}(\sigma)=\ell}} \|A|_{\sigma\times\tau} - \mathcal{T}_{p'}(A|_{\sigma\times\tau})\|_2 \\
&= C_{\mathrm{sparse}} \sum_{\ell=0}^{\operatorname{depth}(\mathbb{P})} \max_{\substack{(\sigma,\tau)\in\mathbb{P}_{\mathrm{far}}\\ \operatorname{level}(\sigma)=\ell}} \sigma_{p'+1}(A|_{\sigma\times\tau}) \\
&\le C_{\mathrm{sparse}}(\operatorname{depth}(\mathbb{P})+1)\,\varepsilon\,\|A\|_2,
\end{aligned} \tag{3.28}$$

where we have used (3.25) and $\sigma_1(A|_{\sigma\times\tau}) = \|A|_{\sigma\times\tau}\|_2 \le \|A\|_2$. ■

The following algorithm is used to compress a far field block $(\sigma,\tau) \in \mathbb{P}_{\mathrm{far}}$ of the matrix A.

Algorithm 3.17 (Recompression of Far Field Block).

`function CompressPfar(var` A^{comp}`,`A`,`σ`,`τ`,`ε`)`
Compute singular value decomposition of $A|_{\sigma\times\tau}$
Seek for minimal $p' \in \{1,\dots,p-1\}$ with (3.25).
`if` p' exists
  Define $A^{\mathrm{comp}}|_{\sigma\times\tau} := \mathcal{T}_{p'}(A|_{\sigma\times\tau})$ and store it in factorized form.
`else`
  Define $A^{\mathrm{comp}}|_{\sigma\times\tau} := A|_{\sigma\times\tau}$.
`end`

Lemma 3.18. *For $(\sigma,\tau) \in \mathbb{P}_{\text{far}}$, Algorithm 3.17 needs*

$$6p^2(|\sigma| + |\tau|) + 23p^3 + p|\sigma| + p \tag{3.29}$$

operations.

Proof. According to Corollary 3.8, the compression onto rank p' has complexity $6p^2(|\sigma| + |\tau|) + 23p^3 + |\sigma|p'$. Moreover, it takes at most p operations to check whether p' exists and eventually to compute it. ■

For a near field block, we follow the same strategy. However, to compute the singular value decomposition efficiently, we have to compute a factorization of $A|_{\sigma\times\tau}$ if the near field block $(\sigma,\tau) \in \mathbb{P}_{\text{near}}$ is "too large". We note that, for $(\sigma,\tau) \in \mathbb{P}_{\text{near}}$, there holds either $|\sigma| \le C_{\text{leaf}}$ or $|\tau| \le C_{\text{leaf}}$.

Algorithm 3.19 (Recompression of Near Field Block).

`function CompressPnear(var` A^{comp}`,`A`,`σ`,`τ`,`ε`)`
`if` $|\sigma| \le C_{\text{leaf}}$ `and` $|\tau| \le C_{\text{leaf}}$
  Compute singular value decomposition of $A|_{\sigma\times\tau}$ directly.
`else`
  `if` $|\tau| \le |\sigma|$ `/*` i.e. $|\tau| \le C_{\text{leaf}} < |\sigma|$ `*/`
    Compute reduced QR-factorization $A|_{\sigma\times\tau} = QR$ with $Q \in \mathbb{K}^{|\sigma|\times|\tau|}$ and $R \in \mathbb{K}^{|\tau|\times|\tau|}$.
    Define $V := Q$ and $W := R^H$.
    Compute singular value decomposition of $A|_{\sigma\times\tau} = VW^H$.
  `else /*` i.e. $|\sigma| \le C_{\text{leaf}} < |\tau|$ `*/`
    Compute reduced QR-factorization $(A|_{\sigma\times\tau})^T = QR$ with $Q \in \mathbb{K}^{|\tau|\times|\sigma|}$ and $R \in \mathbb{K}^{|\sigma|\times|\sigma|}$.
    Define $V := R^T$ and $W := \overline{Q}$ and observe $A|_{\sigma\times\tau} = R^TQ^T = VW^H$.
    Compute singular value decomposition of $A|_{\sigma\times\tau} = VW^H$.
  `end`
`end`
Seek for minimal $p' \in \{1,\dots,p\}$ with (3.25).
`if` p' exists
  Define $A^{\text{comp}}|_{\sigma\times\tau} := \mathcal{T}_{p'}(A|_{\sigma\times\tau})$ and store it in factorized form.
`else`
  `if` $\operatorname{rank}(A|_{\sigma\times\tau}) \le p$ `/*` i.e. $\operatorname{rank}(A|_{\sigma\times\tau}) = p$ `*/`
    Define $A^{\text{comp}}|_{\sigma\times\tau} := A|_{\sigma\times\tau}$ but store it in factorized form.
  `else`
    Define $A^{\text{comp}}|_{\sigma\times\tau} := A|_{\sigma\times\tau}$.
  `end`
`end`

Lemma 3.20. *For $(\sigma,\tau) \in \mathbb{P}_{\text{near}}$, Algorithm 3.19 needs less than*

$$12\max\{C_{\text{leaf}}^2, p\}(|\sigma| + |\tau|) + C_{\text{leaf}}^2 + 23C_{\text{leaf}}^3 + p \tag{3.30}$$

operations.

Proof. We distinguish the three cases to compute the singular value decomposition:

First, in case of $\max\{|\sigma|, |\tau|\} \leq C_{\text{leaf}}$, the direct computation of the singular value decomposition takes less than $21C_{\text{leaf}}^3$ operations. Second, we consider the case $|\tau| \leq C_{\text{leaf}} < |\sigma|$:

- Computation of QR-factorization $A|_{\sigma\times\tau} = QR$ needs $4|\sigma||\tau|^2 \leq 4|\sigma|C_{\text{leaf}}^2$ operations.
- Transposing $W := R^H$ needs less than $|\tau|^2 \leq C_{\text{leaf}}^2$ operations.
- Computation of singular value decomposition of $A|_{\sigma\times\tau} = QW^H$ needs less than $6|\tau|^2(|\sigma|+|\tau|) + 23|\tau|^3 \leq 6C_{\text{leaf}}^2(|\sigma|+|\tau|) + 23C_{\text{leaf}}^3$ operations.

Finally, the case $|\sigma| \leq C_{\text{leaf}} < |\tau|$ has the following computational complexity:

- Computation of $A|_{\sigma\times\tau}^T$ needs $|\sigma||\tau| \leq C_{\text{leaf}}(|\sigma|+|\tau|)$ operations.
- Computation of QR-factorization $A|_{\sigma\times\tau}^T = QR$ needs $4|\tau||\sigma|^2 \leq 4|\tau|C_{\text{leaf}}^2$.
- Transposing $V := R^T$ needs less than $|\sigma|^2 \leq C_{\text{leaf}}^2$ operations.
- Conjugating $W := \overline{Q}$ needs less than $|\tau||\sigma| \leq C_{\text{leaf}}|\tau|$ operations.
- Computation of singular value decomposition of $A|_{\sigma\times\tau}^T = VW^H$ needs less than $6|\sigma|^2(|\sigma|+|\tau|) + 23|\sigma|^3 \leq 6C_{\text{leaf}}^2(|\sigma|+|\tau|) + 23C_{\text{leaf}}^3$ operations.

In any case, the computation of the singular value decomposition of $A|_{\sigma\times\tau}$ is done in less than

$$(2C_{\text{leaf}} + 10C_{\text{leaf}}^2)(|\sigma|+|\tau|) + C_{\text{leaf}}^2 + 23C_{\text{leaf}}^3$$

operations. The determination of p' and the possible computation of $\mathcal{T}_{p'}(A|_{\sigma\times\tau})$ amounts in additional $p + p(|\sigma|+|\tau|)$ operations. ■

Remark. The use of the QR factorization in Algorithm 3.19 might not seem to be the direct way to possibly convert a near-field block $A|_{\sigma\times\tau}$ into a far-field block. The reader, however, may be reminded to the computational complexity of the singular value decomposition, stated in Equation (3.8) on page 5. Therefore, the direct computation of the singular value decomposition of a near field block $A|_{\sigma\times\tau}$ costs $\mathcal{O}(|\sigma|^2|\tau| + |\sigma||\tau|^2 + C_{\text{leaf}}^3)$, which is quadratic (instead of linear) with respect to the larger cluster size $\max\{|\sigma|, |\tau|\}$. □

If we combine both compression steps in the usual way, we are led to a blockwise recompression algorithm for hierarchical matrices. The following recursive algorithm is called by

`CompressHMatrix(`$\emptyset, A, \mathcal{I}, \mathcal{J}, \varepsilon$`)`

and is based on our standard algorithm.

Algorithm 3.21 (Blockwise Recompression of $\mathcal{H}$-Matrices).

```
function CompressHMatrix(var A^comp, A, σ, τ, ε)
if (σ, τ) ∈ P_far
  call CompressPfar(A^comp, A, σ, τ, ε)
elseif sons(σ) ≠ ∅ and sons(τ) ≠ ∅
  for all (σ', τ') ∈ sons(σ) × sons(τ)
    call CompressHMatrix(A^comp, A, σ', τ')
else /* inadmissible block which cannot be split any further */
  call CompressPnear(A^comp, A, σ, τ, ε)
end
```

Proposition 3.22. *Given $A \in \mathcal{H}(p)$ and $\varepsilon > 0$, Algorithm 3.21 needs less than*

$$\left(p+C_{\mathrm{leaf}}^2 + 23r^3\right)|\mathbb{P}| + 12C_{\mathrm{sparse}}(\mathrm{depth}(\mathbb{P})+1)(m+n)r^2 = \mathcal{O}(p^3N + p^2N\log N) \quad (3.31)$$

operations to construct $A^{\mathrm{comp}} \in \mathcal{H}(p)$, where $r := \max\{C_{\mathrm{leaf}}, p\}$ and $N := m+n$. Moreover, there holds

$$\frac{\|A - A^{\mathrm{comp}}\|_2}{\|A\|_2} \leq C_{\mathrm{sparse}}(\mathrm{depth}(\mathbb{P})+1)\,\varepsilon. \quad (3.32)$$

Proof. The error estimate (3.32) has been proven in Lemma 3.16. The numerical effort is less than

$$\begin{aligned}
&\sum_{(\sigma,\tau)\in\mathbb{P}_{\mathrm{far}}} \left[7p^2(|\sigma|+|\tau|) + 23p^3 + p\right] \\
&\quad + \sum_{(\sigma,\tau)\in\mathbb{P}_{\mathrm{near}}} \left[12\max\{C_{\mathrm{leaf}},p\}^2(|\sigma|+|\tau|) + C_{\mathrm{leaf}}^2 + 23C_{\mathrm{leaf}}^3 + p\right] \\
&\leq \left[C_{\mathrm{leaf}}^2 + 23\max\{p, C_{\mathrm{leaf}}\}^3 + p\right]|\mathbb{P}| + 12\max\{C_{\mathrm{leaf}},p\}^2 \sum_{(\sigma,\tau)\in\mathbb{P}} (|\sigma|+|\tau|),
\end{aligned}$$

from which we conclude the proof with the usual techniques. ■

Remark. Proposition 3.22 indicates to choose $\varepsilon = \left[C_{\mathrm{sparse}}(\mathrm{depth}(\mathbb{P})+1)\right]^{-1}\varepsilon_{\mathrm{tol}}$, which leads to a guaranteed relative error less than $\varepsilon_{\mathrm{tol}} > 0$. In practice, one observes that the choice $\varepsilon = \varepsilon_{\mathrm{tol}}$ is sufficient. □

Remark. If one aims for error control with respect to the Frobenius norm, the criterion (3.25) has to be replaced by

$$\sum_{j=p'+1}^{\mathrm{rank}(A|_{\sigma\times\tau})} \sigma_j(A|_{\sigma\times\tau})^2 \leq \varepsilon^2\|A|_{\sigma\times\tau}\|_F^2. \quad (3.33)$$

One then obtains

$$\begin{aligned}
\|A - A^{\mathrm{comp}}\|_F^2 = \sum_{(\sigma,\tau)\in\mathbb{P}} \|A|_{\sigma\times\tau} - A^{\mathrm{comp}}|_{\sigma\times\tau}\|_F^2 &\leq \sum_{(\sigma,\tau)\in\mathbb{P}} \|A|_{\sigma\times\tau} - \mathcal{T}_{p'}(A^{\mathrm{comp}}|_{\sigma\times\tau})\|_F^2 \\
&= \sum_{(\sigma,\tau)\in\mathbb{P}} \sum_{j=p'+1}^{\mathrm{rank}(A|_{\sigma\times\tau})} \sigma_j(A|_{\sigma\times\tau})^2 \\
&\leq \varepsilon^2 \sum_{(\sigma,\tau)\in\mathbb{P}} \|A|_{\sigma\times\tau}\|_F^2 \\
&= \varepsilon^2\|A\|_F^2.
\end{aligned}$$

Therefore, there holds $\|A - A^{\mathrm{comp}}\|_F/\|A\|_F \leq \varepsilon$. Note that it takes $\mathcal{O}(p)$ operations to determine p', i.e., we just have to replace the "single p" in the complexity estimate by $\mathcal{O}(p)$. □

3.4.2 Coarsening of the Block Partitioning

In this section, we aim to coarsen the block partitioning of an already computed $\mathcal{H}$-matrix $A \in \mathcal{H}(p)$. More precisely, if possible, we want to replace four admissible son blocks $(\sigma_i, \tau_j) \in \mathbb{P}_{\text{far}}$ by its father. The main observation is the following remark, which states that the singular value decomposition of the father block can be computed efficiently.

Remark. Let $(\sigma, \tau) \in \mathbb{T}_{\mathcal{I}} \times \mathbb{T}_{\mathcal{J}}$ have four admissible sons $(\sigma_i, \tau_j) \in \mathbb{P}_{\text{far}}$, for $i, j \in \{1, 2\}$, and assume that

$$A|_{\sigma_i \times \tau_j} = V_{ij} W_{ij}^H \text{ with matrices } V_{ij} \in \mathbb{K}^{|\sigma_i| \times p} \text{ and } W_{ij} \in \mathbb{K}^{|\tau_j| \times p}.$$

Defining the matrices

$$V := \begin{pmatrix} V_{11} & 0 & V_{12} & 0 \\ 0 & V_{21} & 0 & V_{22} \end{pmatrix} \in \mathbb{K}^{|\sigma| \times 4p} \quad \text{and} \quad W := \begin{pmatrix} W_{11} & W_{21} & 0 & 0 \\ 0 & 0 & W_{12} & W_{22} \end{pmatrix} \in \mathbb{K}^{|\tau| \times 4p},$$

we observe that

$$VW^H = \begin{pmatrix} V_{11}W_{11}^H & V_{12}W_{12}^H \\ V_{21}W_{21}^H & V_{12}W_{22}^H \end{pmatrix} = \begin{pmatrix} A|_{\sigma_1 \times \tau_1} & A|_{\sigma_1 \times \tau_2} \\ A|_{\sigma_2 \times \tau_1} & A|_{\sigma_2 \times \tau_2} \end{pmatrix} = A|_{\sigma \times \tau}.$$

Therefore, the father block $A|_{\sigma \times \tau}$ has (at most) rank $4p$, and the singular value decomposition of $A|_{\sigma \times \tau}$ can be computed in (less than)

$$6(4p)^2(|\sigma| + |\tau|) + 23(4p)^3 = 96p^2(|\sigma| + |\tau|) + 1472p^3$$

arithmetic operations. □

We now state the recursive algorithm, which tries to coarsen the block partitioning. The function is called by

`CoarsenHMatrix(`$\emptyset, A, \mathcal{I}, \mathcal{J}, \varepsilon$`)`

where ε is the parameter which steers the coarsening. The coarsening criterion (3.25) is the same as for the blockwise recompression. To abbreviate notation, we write

$$\text{sons}(\sigma \times \tau) := \begin{cases} \text{sons}(\sigma) \times \text{sons}(\tau), & \text{if sons}(\sigma) \neq \emptyset \neq \text{sons}(\tau), \\ \emptyset, & \text{else.} \end{cases}$$

Algorithm 3.23 (Coarsening of Block Partitioning).

```
function CoarsenHMatrix(var A^comp,A,σ,τ,ε)
if sons(σ × τ) = ∅
   Define A^comp|_{σ×τ} := A|_{σ×τ}.
   return
end
/* Guarantee that all sons are admissible */
for all (σ',τ') ∈ sons(σ × τ)
   call CoarsenHMatrix(A^comp,A,σ',τ',ε)
   if (σ',τ') ∉ P^comp_far
      return
   end
end
/* Try to coarsen block partitioning */
Compute singular value decomposition of A^comp|_{σ×τ}.
Seek for minimal p' ∈ {1,...,p} such that (3.25) holds for A^comp|_{σ×τ}.
if p' exists
   for all (σ',τ') ∈ sons(σ × τ)
      Remove (σ',τ') from P^comp_far and delete son blocks A^comp|_{σ'×τ'}.
   end
   Add (σ,τ) to P_far and store A^comp|_{σ×τ} := T_{p'}(A^comp|_{σ×τ}) in factorized form.
end
```

Proposition 3.24. *Given $A \in \mathcal{H}(p, \mathbb{A})$ and $\varepsilon > 0$, Algorithm 3.23 needs less than*

$$\begin{aligned} &C_{\text{sparse}}\text{depth}(\mathbb{P})(\text{depth}(\mathbb{P})+1)(96p^2+p)(m+n) + (1472p^3+p)|\mathbb{P}_{\text{far}}| \\ &\quad = \mathcal{O}(p^2 N \log^2 N + p^3 N) \end{aligned} \tag{3.34}$$

operations to construct $A^{\text{comp}} \in \mathcal{H}(p, \mathbb{A}^{\text{comp}})$, where $N := m + n$. Moreover, there holds

$$\frac{\|A - A^{\text{comp}}\|_2}{\|A\|_2} \leq C_{\text{sparse}}\text{depth}(\mathbb{P})(\text{depth}(\mathbb{P})+1)\,\varepsilon. \tag{3.35}$$

Proof. For theoretic reasons, the recursive algorithm is stated inductively.

- We define $(A^{(0)}, \mathbb{P}^{(0)}) := (A, \mathbb{P})$.
- For $j = 1, \ldots, \text{depth}(\mathbb{P})$ and given $(A^{(j-1)}, \mathbb{P}^{(j-1)})$, the next pairing $(A^{(j)}, \mathbb{P}^{(j)})$ is obtained from $(A^{(j-1)}, \mathbb{P}^{(j-1)})$ as follows: We only consider and allow blocks $(\sigma, \tau) \in \mathbb{T}_{\mathcal{I}} \times \mathbb{T}_{\mathcal{J}}$ with $\text{level}(\sigma \times \tau) = \text{depth}(\mathbb{P}) - j$ for coarsening.

Clearly, $A^{\text{comp}} = A^{(\text{depth}(\mathbb{P}))}$. Moreover, the corresponding block trees $\mathbb{B}^{(j)}$ are subtrees of $\mathbb{B}^{(0)} = \mathbb{B}$, which implies $C_{\text{sparse}} := C_{\text{sparse}}(\mathbb{B}) \geq C_{\text{sparse}}(\mathbb{B}^{(j)}$. Therefore, the error can be

estimated by use of the triangle inequality:

$$\begin{aligned}\|A - A^{\mathrm{comp}}\|_2 &\leq \sum_{j=1}^{\mathrm{depth}(\mathbb{P})} \|A^{(j-1)} - A^{(j)}\|_2 \\ &\leq C_{\mathrm{sparse}} \sum_{j=1}^{\mathrm{depth}(\mathbb{P})} \sum_{\ell=0}^{\mathrm{depth}(\mathbb{P}^{(j)})} \max_{\substack{(\sigma,\tau)\in\mathbb{P}^{(j)} \\ \mathrm{level}(\sigma\times\tau)=\ell}} \|A^{(j-1)}|_{\sigma\times\tau} - A^{(j)}|_{\sigma\times\tau}\|_2.\end{aligned}$$

According to the definition of $A^{(j)}$, there holds

$$\begin{aligned}\|A^{(j-1)}|_{\sigma\times\tau} - A^{(j)}|_{\sigma\times\tau}\|_2 \leq \|A^{(j-1)}|_{\sigma\times\tau} - \mathcal{T}_{p'}(A^{(j-1)}|_{\sigma\times\tau})\|_2 &= \sigma_{p'+1}(A^{(j-1)}|_{\sigma\times\tau}) \\ &\leq \varepsilon\sigma_1(A^{(j-1)}|_{\sigma\times\tau}).\end{aligned}$$

Together with

$$\sigma_1(A^{(j-1)}|_{\sigma\times\tau}) = \sigma_1(A|_{\sigma\times\tau}) = \|A|_{\sigma\times\tau}\|_2 \leq \|A\|_2,$$

we obtain the claimed error estimate from $\mathrm{depth}(\mathbb{P}^{(j)}) \leq \mathrm{depth}(\mathbb{P})$. For the computational complexity, we consider one step from $A^{(j-1)}$ to $A^{(j)}$. The number of arithmetic operations is less than

$$\sum_{\substack{(\sigma,\tau)\in\mathbb{T}_{\mathcal{I}}\times\mathbb{T}_{\mathcal{J}} \\ \mathrm{sons}(\sigma\times\tau)\in\mathbb{P}_{\mathrm{far}}^{(j-1)}}} \Big(96p^2(|\sigma| + |\tau|) + 1472p^3 + |\sigma|p + p\Big),$$

where we took the computation of the singular value decomposition into account as well as the determination of p' and the storage of $\mathcal{T}_{p'}(A|_{\sigma\times\tau})$ in factorized form. This sum is bounded from above by

$$\begin{aligned}&\sum_{(\sigma',\tau')\in\mathbb{P}_{\mathrm{far}}^{(j-1)}} \Big(96p^2(|\sigma'| + |\tau'|) + |\sigma'|p + 1472p^3 + p\Big) \\ &\quad\leq C_{\mathrm{sparse}}(\mathrm{depth}(\mathbb{P}^{(j-1)}) + 1)(96p^2 + p)(m+n) + (1472p^3 + p)|\mathbb{P}_{\mathrm{far}}^{(j-1)}| \\ &\quad\leq C_{\mathrm{sparse}}(\mathrm{depth}(\mathbb{P}) + 1)(96p^2 + p)(m+n) + (1472p^3 + p)|\mathbb{P}_{\mathrm{far}}|\end{aligned}$$

by use of the usual technique. ■

Remark. Note that the complexity estimate and the error estimate in Proposition 3.24 are very pessimistic. In practice, one observes

$$\frac{\|A - A^{\mathrm{comp}}\|_2}{\|A\|_2} \leq \varepsilon_{\mathrm{tol}}$$

for $\varepsilon = \varepsilon_{\mathrm{tol}}$ instead of $\varepsilon = \big(C_{\mathrm{sparse}}\mathrm{depth}(\mathbb{P})(\mathrm{depth}(\mathbb{P}) + 1)\big)^{-1}\varepsilon_{\mathrm{tol}}$. □

3.4.3 Full Recompression of $\mathcal{H}$-Matrix

The full recompression of a hierarchical matrix consists of blockwise recompression of the admissible blocks as well as simultaneous coarsening of the block partitioning.

Algorithm 3.25 (Full Recompression of $\mathcal{H}$-Matrices).

```
function FullCompressHMatrix(var A^comp, A, σ, τ, ε_1, ε_2)
if (σ, τ) ∈ P_far
   call CompressPfar(A^comp, A, σ, τ, ε_1)
elseif sons(σ × τ) ≠ ∅
   for all (σ', τ') ∈ sons(σ × τ)
      call FullCompressHMatrix(A^comp, A, σ', τ', ε_1, ε_2)
      if (σ', τ') ∉ P_far^comp
         return
      end
   end
   Compute singular value decomposition of A^comp|_{σ×τ}.
   Seek for minimal p' ∈ {1, ..., p} such that (3.25) holds for A^comp|_{σ×τ} and ε = ε_2.
   if p' exists
      for all (σ', τ') ∈ sons(σ × τ)
         Remove (σ', τ') from P_far^comp and delete son blocks A^comp|_{σ'×τ'}.
      end
      Add (σ, τ) to P_far and store A^comp|_{σ×τ} := T_{p'}(A^comp|_{σ×τ}) in factorized form.
   end
else /* inadmissible block which cannot be split any further */
   call CompressPnear(A^comp, A, σ, τ, ε_1)
end
```

For the computational complexity of Algorithm 3.25 one simply has to add the complexity of each part, namely blockwise recompression and coarsening of the block partitioning. We stress the error control

$$\frac{\|A - A^{\mathrm{comp}}\|_2}{\|A\|_2} \leq C_{\mathrm{sparse}}(\mathrm{depth}(\mathbb{P}) + 1)\big[\varepsilon_1 + \mathrm{depth}(\mathbb{P})\varepsilon_2\big]. \tag{3.36}$$

To prove this, let A^{block} denote the matrix obtained from blockwise recompression of A, i.e. A^{comp} is obtained by coarsening the block structure of A^{block}. The triangle inequality proves

$$\|A - A^{\mathrm{comp}}\|_2 \leq \|A - A^{\mathrm{block}}\|_2 + \|A^{\mathrm{block}} - A^{\mathrm{comp}}\|_2.$$

The first term is estimated by Proposition 3.22. The second term is estimated as in Proposition 3.24, where one uses $\sigma_1(A^{\mathrm{block}}|_{\sigma\times\tau}) = \sigma_1(A|_{\sigma\times\tau}) \leq \|A\|_2$ to get $\|A\|_2$ instead of only $\|A^{\mathrm{block}}\|_2$ in the upper bound.

3.4.4 On-The-Fly-Recompression

Usually, the assembly and the recompression of an $\mathcal{H}$-matrix is coupled since the total computing time for the assembly is usually higher than the time for the recompression. In this case, an assembled matrix block $A|_{\sigma\times\tau}$ is, at once, replaced by the recompressed matrix block.

Algorithm 3.26 (On-The-Fly-Recompression).

```
function BuildCompressedHMatrix(var A,σ,τ,ε1,ε2)
if (σ,τ) is admissible
   Compute A|_{σ×τ} = VW^H in factorized form.
   call CompressPfar(A,A,σ,τ,ε1)
elseif sons(σ × τ) ≠ ∅
   for all (σ',τ') ∈ sons(σ) × sons(τ)
      call BuildCompressedHMatrix(A,σ',τ',ε1,ε2)
      if (σ',τ') ∉ P_far
         return
      end
   end
   Compute singular value decomposition of A|_{σ×τ}.
   Seek for minimal p' ∈ {1,...,p} such that (3.25) with ε = ε2.
   if p' exists
      Add (σ,τ) to P_far and store A|_{σ×τ} := T_{p'}(A|_{σ×τ}) in factorized form.
      for all (σ',τ') ∈ sons(σ × τ)
         Remove (σ',τ') from P_far and delete son blocks A|_{σ'×τ'}.
   end
else /* inadmissible block which cannot be split any further */
   Compute full matrix block A|_{σ×τ}.
   call CompressPnear(A,A,σ,τ,ε1)
end
```

3.5 $\mathcal{H}$-Addition (19.05.2009)

Given an $\mathcal{H}$-tree $\mathbb{B}$ for $\mathcal{I} \times \mathcal{J}$, an admissibility condition $\mathbb{A}$, and $\mathcal{H}$-matrices $A, B \in \mathcal{H}(p) := \mathcal{H}(\mathbb{B}, \mathbb{A}, p)$, we consider the matrix sum $A + B$. Note that, in general, $A + B$ does not belong to $\mathcal{H}(p)$ but only to $\mathcal{H}(2p)$ as can be seen from the following elementary example for rank-1-matrices:

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

We therefore define the $\mathcal{H}$-addition

$$\oplus_p : \mathcal{H}(p) \times \mathcal{H}(p) \to \mathcal{H}(p), \quad A \oplus_p B := \mathcal{T}_{\mathcal{H}(p)}(A + B). \tag{3.37}$$

Note that $A \oplus_p B$ is the best approximation of $A + B$ with respect to the Frobenius norm. The following recursive algorithm, called by

```
addHMatrices(var C,A,B,p,I,J)
```

computes the sum $C := A \oplus_p B$.

Algorithm 3.27 (Addition of $\mathcal{H}$-Matrices with fixed local rank).

```
function AddHMatrices(var C,A,B,σ,τ,p)
if (σ,τ) ∈ ℙ_far
   C|_{σ×τ} := 𝒯_p(A|_{σ×τ} + B|_{σ×τ})
elseif sons(σ × τ) ≠ ∅
   for all (σ',τ') ∈ sons(σ × τ)
      call AddHMatrices(C,A,B,σ',τ',p)
else /* inadmissible block which cannot be split any further */
   C|_{σ×τ} := A|_{σ×τ} + B|_{σ×τ}
end
```

We stress that we don't perform the addition in the far field as stated in the algorithm. Instead, we write $A|_{\sigma\times\tau} = V_1 W_1^H$ and $B|_{\sigma\times\tau} = V_2 W_2^H$ with matrices $V_j \in \mathbb{K}^{|\sigma|\times p}$ and $W_j \in \mathbb{K}^{|\tau|\times p}$ and define $V := (V_1, V_2) \in \mathbb{K}^{|\sigma|\times 2p}$ and $W := (W_1, W_2) \in \mathbb{K}^{|\tau|\times 2p}$. Then, $A|_{\sigma\times\tau} + B|_{\sigma\times\tau} = VW^H$, i.e. the *exact addition* of far field blocks essentially is a memory copy. Finally, $C|_{\sigma\times\tau} = \mathcal{T}_p(VW^H)$ is computed with the help of the singular value decomposition.

Proposition 3.28. *The $\mathcal{H}$-addition of two matrices $A, B \in \mathcal{H}(p)$ needs less than*

$$\begin{aligned} &184p^3|\mathbb{P}_{\text{far}}| + C_{\text{sparse}}(\text{depth}(\mathbb{P}) + 1)\max\{24p^2 + p, C_{\text{leaf}}\}(m+n) \\ &\quad = \mathcal{O}(p^3 N + p^2 N \log N) \end{aligned} \tag{3.38}$$

arithmetic operations, where $N = m + n$.

Proof. For a near field block $\sigma\times\tau \in \mathbb{P}_{\text{near}}$, the addition is done exactly in $|\sigma||\tau| \le C_{\text{leaf}}(|\sigma|+|\tau|)$ operations. For a far field block, we need

$$6(2p)^2(|\sigma| + |\tau|) + 23(2p)^3 + |\sigma|p \le (24p^2 + p)(|\sigma| + |\tau|) + 184p^3$$

operations to compute and store $\mathcal{T}_p(A|_{\sigma\times\tau} + B|_{\sigma\times\tau})$ in factorized form. The usual techniques conclude the proof. ■

Remark. The recompression of $C := A \oplus_p B \in \mathcal{H}(p)$ can be done on-the-fly in Algorithm 3.27. We note that $\mathcal{H}$-addition can be arbitrarily bad, e.g., for rank-1-matrices

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \oplus_1 \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \mathcal{T}_1 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

Alternatively to the above algorithm, one can therefore compute a recompressed matrix $C := A \oplus_\varepsilon B \in \mathcal{H}(2p)$ such that

$$\frac{\|(A+B) - C\|_2}{\|A+B\|_2} \le \varepsilon_{\text{tol}} \tag{3.39}$$

up to a chosen tolerance $\varepsilon_{\text{tol}} > 0$. This is called **adaptive $\mathcal{H}$-addition** in the literature. Modifications are obvious from the previous section and thus omitted. □

Remark. Clearly, the $\mathcal{H}$-addition works for $A, B \in \mathcal{H}(\mathbb{B}, \mathbb{A})$ with different local ranks p_A and p_B, respectively. Finally, the algorithm can easily be adopted to deal with recompressed $\mathcal{H}$-matrices: In practice, we compute $A, B \in \mathcal{H}(\mathbb{B}, \mathbb{A})$, but recompression leads to $A \in \mathcal{H}(\mathbb{B}_A^{\text{comp}},$

$\mathbb{A}_A^{\text{comp}}$) and $B \in \mathcal{H}(\mathbb{B}_B^{\text{comp}}, \mathbb{A}_B^{\text{comp}})$, where $\mathbb{B}_A^{\text{comp}}$ and $\mathbb{B}_B^{\text{comp}}$ are two (in general different) subtrees of $\mathbb{B}$, and $\mathbb{A}_A^{\text{comp}}$ and $\mathbb{A}_B^{\text{comp}}$ are two (in general different) weakened admissibility conditions. Therefore, one has to build the common supertree $\mathbb{B}_C \leq \mathbb{B}$ of $\mathbb{B}_A^{\text{comp}}$ and $\mathbb{B}_B^{\text{comp}}$, and the blocks of A und B must be converted, accordingly. The nasty and technical details are left to the reader. □

3.6 $\mathcal{H}$-Multiplication (25.05.2009)

Compared with the $\mathcal{H}$-addition, the $\mathcal{H}$-multiplication is a rather complicated operation. The reason for this is that matrix-matrix multiplication reads blockwise

$$(AB)|_{\rho\times\tau} = \sum_{j\in\mathcal{J}} A|_{\rho\times\{j\}} B|_{\{j\}\times\tau} \quad \text{for all } \rho \subseteq \mathcal{I} \text{ and } \tau \subseteq \mathcal{K}. \tag{3.40}$$

The inner sum affects the block partitioning, i.e. the block partitioning can change completely under the matrix-matrix multiplication. We give three elementary examples, where $a_{jk} = 0$ may denote a matrix block with a coarse structure and $a_{jk} \neq 0$ denotes a matrix block with a finer structure:

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix},$$

i.e. the block structure of the product is coarser than the structure of the factors. Of course, this is not a problem since we may deal with the finer structure then. Second,

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$$

i.e. the block structure of the product is finer than the structure of the factors. Therefore, the storage requirements increase when compared to each factor. Finally,

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

i.e. the block structure can even change totally. In a first step, we therefore define a block tree which is capable to store the exact matrix-matrix product.

Throughout this section, we assume that $\mathbb{B}_{\mathcal{I}\times\mathcal{J}} := \mathbb{B}(\mathbb{T}_{\mathcal{I}}, \mathbb{T}_{\mathcal{J}}, \mathbb{A}_{\mathcal{I}\times\mathcal{J}})$ and $\mathbb{B}_{\mathcal{J}\times\mathcal{K}} := \mathbb{B}(\mathbb{T}_{\mathcal{J}}, \mathbb{T}_{\mathcal{K}}, \mathbb{A}_{\mathcal{J}\times\mathcal{K}})$ are block trees that are built from given cluster trees $\mathbb{T}_{\mathcal{I}}$, $\mathbb{T}_{\mathcal{J}}$, and $\mathbb{T}_{\mathcal{K}}$ and associated admissibility conditions $\mathbb{A}_{\mathcal{I}\times\mathcal{J}}$ and $\mathbb{A}_{\mathcal{J}\times\mathcal{K}}$, respectively.

3.6.1 Product Tree Based Estimates

Definition. For the given block trees $\mathbb{B}_{\mathcal{I}\times\mathcal{J}}$ and $\mathbb{B}_{\mathcal{J}\times\mathcal{K}}$, we define the **product tree** $\mathbb{B}_{\mathcal{I}\times\mathcal{J}} \cdot \mathbb{B}_{\mathcal{J}\times\mathcal{K}}$ by $\operatorname{root}(\mathbb{B}_{\mathcal{I}\times\mathcal{J}} \cdot \mathbb{B}_{\mathcal{J}\times\mathcal{K}}) := \mathcal{I} \times \mathcal{K}$ and

$$\operatorname{sons}(\rho\times\tau) := \left\{ \rho'\times\tau' \,\middle|\, (\rho',\tau') \in \mathbb{T}_{\mathcal{I}}\times\mathbb{T}_{\mathcal{K}}, \exists \sigma, \sigma' \in \mathbb{T}_{\mathcal{J}} \quad \rho'\times\sigma' \in \operatorname{sons}(\rho\times\sigma), \sigma'\times\tau' \in \operatorname{sons}(\sigma\times\tau) \right\},$$

for all $\rho\times\tau \in \mathbb{B}_{\mathcal{I}\times\mathcal{J}}\cdot\mathbb{B}_{\mathcal{J}\times\mathcal{K}}$. Moreover, all leaves of the product tree are regarded as admissible.

As a first observation, we note that the product tree is an $\mathcal{H}$-tree and that depth and sparsity can be estimated by the corresponding quantities of $\mathbb{B}_{\mathcal{I}\times\mathcal{J}}$ and $\mathbb{B}_{\mathcal{J}\times\mathcal{K}}$.

Lemma 3.29. *The product tree is an $\mathcal{H}$-tree for the index set $\mathcal{I}\times\mathcal{K}$, and there holds*

$$\operatorname{depth}(\mathbb{B}_{\mathcal{I}\times\mathcal{J}}\cdot\mathbb{B}_{\mathcal{J}\times\mathcal{K}}) \leq \min\{\operatorname{depth}(\mathbb{B}_{\mathcal{I}\times\mathcal{J}}),\operatorname{depth}(\mathbb{B}_{\mathcal{J}\times\mathcal{K}})\}. \tag{3.41}$$

Moreover, the sparsity constant of the product tree satisfies

$$C_{\text{sparse}}(\mathbb{B}_{\mathcal{I}\times\mathcal{J}}\cdot\mathbb{B}_{\mathcal{J}\times\mathcal{K}}) \leq C_{\text{sparse}}(\mathbb{B}_{\mathcal{I}\times\mathcal{J}})C_{\text{sparse}}(\mathbb{B}_{\mathcal{J}\times\mathcal{K}}) \tag{3.42}$$

Proof. To prove that $\mathbb{B} := \mathbb{B}_{\mathcal{I}\times\mathcal{J}}\cdot\mathbb{B}_{\mathcal{J}\times\mathcal{K}}$ is an $\mathcal{H}$-tree, let $\rho\times\tau\in\mathbb{B}$ with $\operatorname{sons}(\rho\times\tau)\neq\emptyset$. We have to show that

- $|\operatorname{sons}(\rho\times\tau)|\geq 2$,
- $\rho'\times\tau'\neq\emptyset$ for all $\rho'\times\tau'\in\operatorname{sons}(\rho\times\tau)$,
- $\rho\times\tau$ is the disjoint union of its sons.

All of which is easily seen from the definition of the sons-relation and the respective properties of $\mathbb{B}_{\mathcal{I}\times\mathcal{J}}$ and $\mathbb{B}_{\mathcal{J}\times\mathcal{K}}$. The estimate for the depth of $\mathbb{B}$ follows directly from the definition of the sons-relation. It thus only remains to estimate the sparsity constant: Given $\rho\in\mathbb{T}_{\mathcal{I}}$, there holds

$$\begin{aligned}\big\{\tau\in\mathbb{T}_{\mathcal{K}}\,\big|\,\rho\times\tau\in\mathbb{B}\big\} &\subseteq \big\{\tau\in\mathbb{T}_{\mathcal{K}}\,\big|\,\exists\sigma\in\mathbb{T}_{\mathcal{J}}\quad \rho\times\sigma\in\mathbb{B}_{\mathcal{I}\times\mathcal{J}},\,\sigma\times\tau\in\mathbb{B}_{\mathcal{J}\times\mathcal{K}}\big\}\\ &= \bigcup_{\substack{\sigma\in\mathbb{T}_{\mathcal{J}}\\ \rho\times\sigma\in\mathbb{B}_{\mathcal{I}\times\mathcal{J}}}} \big\{\tau\in\mathbb{T}_{\mathcal{K}}\,\big|\,\sigma\times\tau\in\mathbb{B}_{\mathcal{J}\times\mathcal{K}}\big\}.\end{aligned}$$

Consequently, we obtain

$$\begin{aligned}\big|\big\{\tau\in\mathbb{T}_{\mathcal{K}}\,\big|\,\rho\times\tau\in\mathbb{B}\big\}\big| &\leq \sum_{\substack{\sigma\in\mathbb{T}_{\mathcal{J}}\\ \rho\times\sigma\in\mathbb{B}_{\mathcal{I}\times\mathcal{J}}}} \big|\big\{\tau\in\mathbb{T}_{\mathcal{K}}\,\big|\,\sigma\times\tau\in\mathbb{B}_{\mathcal{J}\times\mathcal{K}}\big\}\big|\\ &\leq C_{\text{sparse}}(\mathbb{B}_{\mathcal{I}\times\mathcal{J}})C_{\text{sparse}}(\mathbb{B}_{\mathcal{J}\times\mathcal{K}}).\end{aligned}$$

This concludes the rowwise estimate. For the columnwise estimate the same techniques apply and conclude the proof. ■

Proposition 3.30. *Let $A \in \mathcal{H}(\mathbb{B}_{\mathcal{I}\times\mathcal{J}}, \mathbb{A}_{\mathcal{I}\times\mathcal{J}}, p_A)$ and $B \in \mathcal{H}(\mathbb{B}_{\mathcal{J}\times\mathcal{K}}, \mathbb{A}_{\mathcal{J}\times\mathcal{K}}, p_B)$, and let $\mathbb{B} := \mathbb{B}_{\mathcal{I}\times\mathcal{J}} \cdot \mathbb{B}_{\mathcal{J}\times\mathcal{K}}$ denote the product tree. Then, the exact matrix-matrix product $C := AB$ satisfies $C \in \mathcal{H}(\mathbb{B}, p_C)$, where the local rank p_C can be estimated by*

$$p_C \leq C_{\text{sparse}}(\text{depth}+1)\max\{p_A, p_B, C_{\text{leaf}}\}. \tag{3.43}$$

Moreover, the matrix C can be computed with complexity less than

$$3(\text{depth}+1)^2 C_{\text{sparse}}^3 \max\{p_A, p_B, C_{\text{leaf}}\}^2(|\mathcal{I}|+2|\mathcal{J}|+|\mathcal{K}|) = \mathcal{O}(p^2 N \log^2 N), \tag{3.44}$$

where we define the constants

- $C_{\text{sparse}} := \max\{C_{\text{sparse}}(\mathbb{B}_{\mathcal{I}\times\mathcal{J}}), C_{\text{sparse}}(\mathbb{B}_{\mathcal{J}\times\mathcal{K}})\}$,
- $\text{depth} := \max\{\text{depth}(\mathbb{B}_{\mathcal{I}\times\mathcal{J}}), \text{depth}(\mathbb{B}_{\mathcal{J}\times\mathcal{K}})\}$,
- $N := |\mathcal{I}| + |\mathcal{J}| + |\mathcal{K}|$,
- $p := \max\{p_A, p_B\}$.

The proof is based on the following lemma which provides a partitioning of the index set $\mathcal{J}$ and thus an efficient realization of the sum in (3.40).

Lemma 3.31. *For a leaf $\rho\times\tau$ of the product tree $\mathbb{B} := \mathbb{B}_{\mathcal{I}\times\mathcal{J}} \cdot \mathbb{B}_{\mathcal{J}\times\mathcal{K}}$ and each level $\ell = 0, \ldots, \text{level}(\rho\times\tau)$, we define the set*

$$\mathcal{U}_\ell(\rho\times\tau) := \big\{\sigma \in \mathbb{T}_{\mathcal{J}} \,\big|\, \mathcal{F}^{(\ell)}(\rho)\times\sigma \in \mathbb{B}_{\mathcal{I}\times\mathcal{J}}, \sigma\times\mathcal{F}^{(\ell)}(\tau) \in \mathbb{B}_{\mathcal{J}\times\mathcal{K}},\ \textit{at least one is a leaf}\big\},$$

where $\mathcal{F}^{(\ell)}(\rho) := \widehat{\rho}$ denotes the unique ancestor $\widehat{\rho} \in \mathbb{T}_{\mathcal{I}}$ with $\text{level}(\widehat{\rho}) = \ell$ and $\rho \subseteq \widehat{\rho}$ and where $\mathcal{F}^{(\ell)}(\tau)$ is defined accordingly. Then, the set

$$\bigcup_{\ell=0}^{\text{level}(\rho\times\tau)} \mathcal{U}_\ell(\rho\times\tau) \tag{3.45}$$

is a partition of the index set $\mathcal{J}$. In particular, there holds

$$(AB)|_{\rho\times\tau} = \sum_{j\in\mathcal{J}} A|_{\rho\times\{j\}} B|_{\{j\}\times\tau} \overset{!}{=} \sum_{\ell=0}^{\text{level}(\rho\times\tau)} \sum_{\sigma\in\mathcal{U}_\ell(\rho\times\tau)} A|_{\rho\times\sigma} B|_{\sigma\times\tau}. \tag{3.46}$$

***Proof.* 1. step.** We first show that the sets contained in $\bigcup_\ell \mathcal{U}_\ell(\rho\times\tau)$ are pairwise disjoint: To that end, let $\ell_1 \leq \ell_2$ and $\sigma_1 \in \mathcal{U}_{\ell_1}(\rho\times\tau)$ and $\sigma_2 \in \mathcal{U}_{\ell_2}(\rho\times\tau)$ with $\sigma_1 \cap \sigma_2 \neq \emptyset$. From $\text{level}(\sigma_1) = \ell_1 \leq \ell_2 = \text{level}(\sigma_2)$ and the definition of a cluster tree, we infer $\sigma_2 \subseteq \sigma_1$. Therefore,

$$\mathcal{F}^{(\ell_2)}(\rho)\times\sigma_2 \subseteq \mathcal{F}^{(\ell_1)}(\rho)\times\sigma_1 \quad \text{and} \quad \sigma_2\times\mathcal{F}^{(\ell_2)}(\tau) \subseteq \sigma_1\times\mathcal{F}^{(\ell_1)}(\tau).$$

According to the definition of $\mathcal{U}_{\ell_1}(\rho\times\tau)$, one of the supersets is a leaf. Thus, we have at least one set equality from which we conclude $\sigma_2 = \sigma_1$ as well as $\ell_1 = \ell_2$. Said differently, two sets $\sigma_1, \sigma_2 \in \bigcup_\ell \mathcal{U}_\ell(\rho\times\tau)$ with $\sigma_1 \neq \sigma_2$ satisfy $\sigma_1 \cap \sigma_2 = \emptyset$.

2. step. $\mathcal{J} \subseteq \bigcup_{\ell=0}^{\text{level}(\rho\times\tau)} \bigcup_{\sigma\in\mathcal{U}_\ell(\rho\times\tau)} \sigma$: Let $j \in \mathcal{J}$. We define $\rho_0 := \mathcal{I} = \mathcal{F}^{(0)}(\rho)$, $\sigma_0 := \mathcal{J}$, and $\tau_0 := \mathcal{K} = \mathcal{F}^{(0)}(\tau)$. If neither $\rho_0\times\sigma_0$ nor $\sigma_0\times\tau_0$ is a leaf, they have sons $\rho_1\times\sigma_1$ and

$\widetilde{\sigma}_1 \times \tau_1$ with $j \in \sigma_1 \cap \widetilde{\sigma}_1$. From $\operatorname{level}(\sigma_1) = 1 = \operatorname{level}(\widetilde{\sigma}_1)$, we thus obtain $\sigma_1 = \widetilde{\sigma}_1$. We proceed inductively down to the level ℓ such that either $\rho_\ell \times \sigma_\ell$ or $\sigma_\ell \times \tau_\ell$ is a leaf, which is obviously satisfied for some $\ell \le \operatorname{level}(\rho \times \tau)$. Then, $j \in \sigma_\ell \in \mathcal{U}_\ell(\rho \times \tau)$, which concludes the proof. ■

Proof of Proposition 3.30. We use the representation (3.46)

$$(AB)|_{\rho\times\tau} = \sum_{\ell=0}^{\operatorname{level}(\rho\times\tau)} \sum_{\sigma\in\mathcal{U}_\ell(\rho\times\tau)} A|_{\rho\times\sigma} B|_{\sigma\times\tau}. \tag{3.47}$$

1. step. We first prove that the entire sum in (3.47) has less than $C_{\text{sparse}}(\operatorname{depth}+1)$ addends: Note that the first sum has length $(\operatorname{level}(\rho\times\tau)+1) \le \operatorname{depth}(\mathbb{B})+1 \le \operatorname{depth}+1$. Moreover,

$$\begin{aligned}
\big|\mathcal{U}_\ell(\rho\times\tau)\big| &\le \big|\big\{\sigma\in\mathbb{T}_\mathcal{J} \,\big|\, \mathcal{F}^{(\ell)}(\rho)\times\sigma \in \mathbb{B}_{\mathcal{I}\times\mathcal{J}}\big\}\big| \le C_{\text{sparse}}(\mathbb{B}_{\mathcal{I}\times\mathcal{J}}),\\
\big|\mathcal{U}_\ell(\rho\times\tau)\big| &\le \big|\big\{\sigma\in\mathbb{T}_\mathcal{J} \,\big|\, \sigma\times\mathcal{F}^{(\ell)}(\tau) \in \mathbb{B}_{\mathcal{J}\times\mathcal{K}}\big\}\big| \le C_{\text{sparse}}(\mathbb{B}_{\mathcal{J}\times\mathcal{K}}),
\end{aligned}$$

Therefore, the second sum has length $|\mathcal{U}_\ell(\rho\times\tau)| \le \min\{C_{\text{sparse}}(\mathbb{B}_{\mathcal{I}\times\mathcal{J}}), C_{\text{sparse}}(\mathbb{B}_{\mathcal{J}\times\mathcal{K}})\} \le C_{\text{sparse}}$.

2. step. To prove (3.43), it remains to estimate prove $\operatorname{rank}(A|_{\rho\times\sigma}B|_{\sigma\times\tau}) \le \max\{p_A, p_B, C_{\text{leaf}}\}$ for $\sigma\in\mathcal{U}_\ell(\rho\times\tau)$: From the definition of $\mathcal{U}_\ell(\rho\times\tau)$, we know that $\mathcal{F}^{(\ell)}(\rho)\times\sigma$ or $\sigma\times\mathcal{F}^{(\ell)}(\tau)$ is a leaf (or even both). Therefore, there holds

- $\operatorname{rank}(A|_{\rho\times\sigma}) \le \operatorname{rank}(A|_{\mathcal{F}^{(\ell)}(\rho)\times\sigma}) \le \max\{p_A, C_{\text{leaf}}\}$
- or $\operatorname{rank}(B|_{\sigma\times\tau}) \le \operatorname{rank}(B|_{\sigma\times\mathcal{F}^{(\ell)}(\tau)}) \le \max\{p_B, C_{\text{leaf}}\}$.

Consequently, $\operatorname{rank}(A|_{\rho\times\sigma}B|_{\sigma\times\tau}) \le \max\{p_A, p_B, C_{\text{leaf}}\}$.

3. step. Estimate (3.44) on computational complexity: To prove that $C = AB$ is an $\mathcal{H}$-matrix with local rank $p_C \le C_{\text{sparse}}(\operatorname{depth}+1)\max\{p_A, p_B, C_{\text{leaf}}\}$, we have to provide a factorization

$$(AB)|_{\rho\times\tau} = U_{\rho\tau}V_{\rho\tau}^H \quad \text{with } U\in\mathbb{K}^{|\rho|\times p_C}, V\in\mathbb{K}^{|\tau|\times p_C} \tag{3.48}$$

for all leaves (ρ,τ) of $\mathbb{B}$. For each addend $C_i := A|_{\rho\times\sigma}B|_{\sigma\times\tau}$ of the sum (3.47), we will provide a factorization

$$A|_{\rho\times\sigma}B|_{\sigma\times\tau} = U_iV_i^H \quad \text{with } U\in\mathbb{K}^{|\rho|\times\widehat{p}_C}, V\in\mathbb{K}^{|\tau|\times\widehat{p}_C},$$

where $\widehat{p}_C := \max\{p_A, p_B, C_{\text{leaf}}\}$. We then may define

$$U_{\rho\tau} := (U_1, U_2, \dots), V_{\rho\tau} := (V_1, V_2, \dots).$$

to provide the desired factorization (3.48). Recall that, by definition of $\mathcal{U}_\ell(\rho\times\tau)$, $\mathcal{F}^{(\ell)}(\rho)\times\sigma$ or $\sigma\times\mathcal{F}^{(\ell)}(\tau)$ is a leaf. Therefore, one of the following cases occurs:

- $\mathcal{F}^{(\ell)}(\rho)\times\sigma$ is admissible, so that $A|_{\mathcal{F}^{(\ell)}(\rho)\times\sigma} = UV^H$ is known in factorized form. Then,

$$A|_{\rho\times\sigma} = U|_{\rho\times\sigma}V^H \quad\text{and}\quad A|_{\rho\times\sigma}B|_{\sigma\times\tau} = U|_{\rho\times\sigma}(V^HB|_{\sigma\times\tau}) = U|_{\rho\times\sigma}(B|_{\sigma\times\tau}^HV)^H =: U_iV_i^H$$

- $\sigma\times\mathcal{F}^{(\ell)}(\tau)$ is admissible, so that $B_{\sigma\times\mathcal{F}^{(\ell)}(\tau)} = UV^H$ is known in factorized form. Then,

$$B|_{\sigma\times\tau} = UV|_{\sigma\times\tau}^H \quad\text{and}\quad A|_{\rho\times\sigma}B|_{\sigma\times\tau} = (A|_{\rho\times\sigma}U)V|_{\sigma\times\tau}^H =: U_iV_i^H$$

- $\mathcal{F}^{(\ell)}(\rho) \times \sigma$ and $\sigma \times \mathcal{F}^{(\ell)}(\tau)$ are inadmissible with $|\sigma| \leq C_{\text{leaf}}$. Then,
$$A|_{\rho\times\sigma} B|_{\sigma\times\tau} = A|_{\rho\times\sigma} (B|_{\sigma\times\tau}^H)^H =: U_i V_i^H.$$
- $\mathcal{F}^{(\ell)}(\rho) \times \sigma$ is inadmissible with $|\rho| \leq |\mathcal{F}^{(\ell)}(\rho)| \leq C_{\text{leaf}}$. Then,
$$A|_{\rho\times\sigma} B|_{\sigma\times\tau} = \boldsymbol{I}\,(B|_{\sigma\times\tau}^H A|_{\rho\times\sigma}^H)^H =: U_i V_i^H.$$
- $\sigma \times \mathcal{F}^{(\ell)}(\tau)$ is inadmissible with $|\tau| \leq |\mathcal{F}^{(\ell)}(\tau)| \leq C_{\text{leaf}}$. Then,
$$A|_{\rho\times\sigma} B|_{\sigma\times\tau} = (A|_{\rho\times\sigma} B|_{\sigma\times\tau})\boldsymbol{I} =: U_i V_i^H.$$

The computation of U_i and V_i thus needs at most $\widehat{p}_C$ matrix-vector multiplication from left with $B|_{\sigma\times\tau}^H$ or $A|_{\rho\times\sigma}$. Per block $(\rho,\tau) \in \mathbb{P}$, the computational complexity is therefore less than

$$\begin{aligned}
&\widehat{p}_C \sum_{\ell=0}^{\text{level}(\rho\times\tau)} \sum_{\sigma\in\mathcal{U}_\ell(\rho\times\tau)} \big(N_{MVM}(A|_{\rho\times\sigma}) + N_{MVM}(B|_{\sigma\times\tau}^H)\big) \\
&\leq 3\,\widehat{p}_C \sum_{\ell=0}^{\text{level}(\rho\times\tau)} \sum_{\sigma\in\mathcal{U}_\ell(\rho\times\tau)} \big(N_{Storage}(A|_{\rho\times\sigma}) + N_{Storage}(B|_{\sigma\times\tau})\big) \\
&= 3\,\widehat{p}_C \big(N_{Storage}(A|_{\rho\times\mathcal{J}}) + N_{Storage}(B|_{\mathcal{J}\times\tau})\big)
\end{aligned}$$

Thus, the full matrix-matrix multiplication has complexity less than

$$\begin{aligned}
&3\,\widehat{p}_C \sum_{(\rho,\tau)\in\mathbb{P}} \big(N_{Storage}(A|_{\rho\times\mathcal{J}}) + N_{Storage}(B|_{\mathcal{J}\times\tau})\big) \\
&= 3\,\widehat{p}_C \sum_{\ell=0}^{\text{depth}(\mathbb{P})} \sum_{\substack{(\rho,\tau)\in\mathbb{P} \\ \text{level}(\rho)=\ell}} \big(N_{Storage}(A|_{\rho\times\mathcal{J}}) + N_{Storage}(B|_{\mathcal{J}\times\tau})\big)
\end{aligned}$$

As usual, we split the second sum to estimate

$$\begin{aligned}
\sum_{\ell=0}^{\text{depth}(\mathbb{P})} \sum_{\substack{(\rho,\tau)\in\mathbb{P} \\ \text{level}(\rho\times\tau)=\ell}} N_{Storage}(A|_{\rho\times\mathcal{J}}) &= \sum_{\ell=0}^{\text{depth}(\mathbb{B})} \sum_{\substack{\rho\in\mathcal{T}_\mathcal{I} \\ \text{level}(\rho)=\ell}} \sum_{\substack{\tau\in\mathcal{T}_\mathcal{K} \\ (\rho,\tau)\in\mathbb{P}}} N_{Storage}(A|_{\rho\times\mathcal{J}}) \\
&\leq C_{\text{sparse}}(\mathbb{B})\,\big(\text{depth}(\mathbb{B}) + 1\big) N_{Storage}(A) \\
&\leq C_{\text{sparse}}^2\,\big(\text{depth} + 1\big) N_{Storage}(A).
\end{aligned}$$

Together with

$$\begin{aligned}
N_{Storage}(A) &\leq C_{\text{sparse}}(\mathbb{B}_{\mathcal{I}\times\mathcal{J}})\big(\text{depth}(\mathbb{B}_{\mathcal{I}\times\mathcal{J}}) + 1\big)\max\{p_A, C_{\text{leaf}}\}\,(|\mathcal{I}| + |\mathcal{J}|) \\
&\leq C_{\text{sparse}}(\text{depth} + 1)\widehat{p}_C\,(|\mathcal{I}| + |\mathcal{J}|)
\end{aligned}$$

and the analogous estimates for the τ-affected sum, we obtain the following upper bound for the computational effort

$$\begin{aligned}
&3\,\widehat{p}_C \sum_{(\rho,\tau)\in\mathbb{P}} \big(N_{Storage}(A|_{\rho\times\mathcal{J}}) + N_{Storage}(B|_{\mathcal{J}\times\tau})\big) \\
&\quad\leq 3\,C_{\text{sparse}}^3\,(\text{depth} + 1)^2\,\widehat{p}_C^2\,\big(|\mathcal{I}| + 2|\mathcal{J}| + |\mathcal{K}|\big).
\end{aligned}$$

This concludes the proof. ■

3.6.2 Estimates for Prescribed Block Tree

In this section, we assume that the $\mathcal{H}$-product matrix $C := AB$ is not to be stored with respect to the product tree $\mathbb{B} = \mathbb{B}_{\mathcal{I}\times\mathcal{J}} \cdot \mathbb{B}_{\mathcal{J}\times\mathcal{K}}$ but with respect to another prescribed block tree $\mathbb{B}_{\mathcal{I}\times\mathcal{K}}$. The so-called idempotency constant C_{id} has been introduced in GRASEDYCK (2001) and GRASEDYCK-HACKBUSCH (2003) in case of $\mathcal{I} = \mathcal{J} = \mathcal{K}$, which explains the nomenclature. It measures how good the leaves of $\mathbb{B}$ are resolved in $\mathbb{B}_{\mathcal{I}\times\mathcal{K}}$, i.e., given a leaf $(\rho,\tau) \in \mathbb{B}_{\mathcal{I}\times\mathcal{K}}$, how many leaves $(\rho',\tau') \in \mathbb{B}$ are contained in (ρ,τ).

Definition. Let $\mathbb{B}_{\mathcal{I}\times\mathcal{J}} := \mathbb{B}(\mathbb{T}_{\mathcal{I}}, \mathbb{T}_{\mathcal{J}})$, $\mathbb{B}_{\mathcal{J}\times\mathcal{K}} := \mathbb{B}(\mathbb{T}_{\mathcal{J}}, \mathbb{T}_{\mathcal{K}})$, and $\mathbb{B}_{\mathcal{I}\times\mathcal{K}} := \mathbb{B}(\mathbb{T}_{\mathcal{I}}, \mathbb{T}_{\mathcal{K}})$ be block cluster trees and $\mathbb{B} := \mathbb{B}_{\mathcal{I}\times\mathcal{J}} \cdot \mathbb{B}_{\mathcal{J}\times\mathcal{K}}$ the induced block tree for $\mathcal{I}\times\mathcal{K}$. For $\rho \in \mathbb{T}_{\mathcal{I}}$ and $\tau \in \mathbb{T}_{\mathcal{K}}$, we define

$$C_{\mathrm{id}}(\rho\times\tau) := \big|\big\{(\rho',\tau') \in S^*(\rho)\times S^*(\tau) \,\big|\, \exists\sigma' \in \mathbb{T}_{\mathcal{J}} \quad (\rho',\sigma') \in \mathbb{B}_{\mathcal{I}\times\mathcal{J}},\, (\sigma',\tau') \in \mathbb{B}_{\mathcal{J}\times\mathcal{K}}\big\}\big|,$$

where $S^*(\rho) := \big\{\rho' \in \mathbb{T}_{\mathcal{I}} \,\big|\, \rho' \subseteq \rho\big\}$ and $S^*(\tau) := \big\{\tau' \in \mathbb{T}_{\mathcal{K}} \,\big|\, \tau' \subseteq \tau\big\}$ denote the sets of successors of ρ and τ, respectively. Finally, we define the **idempotency constant**

$$C_{\mathrm{id}}(\mathbb{B}_{\mathcal{I}\times\mathcal{K}}) := \max\big\{C_{\mathrm{id}}(\rho\times\tau) \,\big|\, (\rho,\tau) \in \mathbb{B}_{\mathcal{I}\times\mathcal{K}} \text{ leaf}\big\}.$$

Remark. In case of $\mathcal{I} = \mathcal{J} = \mathcal{K}$, a block cluster tree $\mathbb{B}$ on $\mathcal{I}\times\mathcal{I}$ is idempotent, if $\mathbb{B} = \mathbb{B}\cdot\mathbb{B}$. Note that idempotency implies $C_{\mathrm{id}}(\mathbb{B}) = 1$, whereas the converse implication fails: The example

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

shows that the given block cluster tree $\mathbb{B}$ might be finer than the product tree $\mathbb{B}\cdot\mathbb{B}$ although $C_{\mathrm{id}}(\mathbb{B}) = 1$. □

Proposition 3.32. *Let $A \in \mathcal{H}(\mathbb{B}_{\mathcal{I}\times\mathcal{J}}, p_A)$ and $B := \mathcal{H}(\mathbb{B}_{\mathcal{J}\times\mathcal{K}}, p_B)$ be given $\mathcal{H}$-matrices and $\mathbb{B}_{\mathcal{I}\times\mathcal{K}}$ be a prescribed block tree. Then, the product matrix $C := AB$ satisfies $C \in \mathcal{H}(\mathbb{B}_{\mathcal{I}\times\mathcal{K}}, p_C)$ for some local rank*

$$p_C \le C_{\mathrm{id}} C_{\mathrm{sparse}}(\mathrm{depth}+1)\max\{p_A, p_B, C_{\mathrm{leaf}}\}. \tag{3.49}$$

Let $\mathbb{A}_{\mathcal{I}\times\mathcal{K}}$ be an associated admissibility condition and $p \le p_C$. It then takes less than

$$\begin{aligned} &10C_{\mathrm{id}}^2 C_{\mathrm{sparse}}^3(\mathrm{depth}+1)^2\max\{p_A,p_B,C_{\mathrm{leaf}}\}^2(|\mathcal{I}|+|\mathcal{J}|+|\mathcal{K}|) \\ &\quad + 23C_{\mathrm{id}}^3 C_{\mathrm{sparse}}^3(\mathrm{depth}+1)^3\max\{p_A,p_B,C_{\mathrm{leaf}}\}^3|\mathbb{P}_{\mathrm{far}}^{(\mathcal{I}\times\mathcal{K})}| \\ &= \mathcal{O}(P^3 N\log^3 N) \end{aligned} \tag{3.50}$$

operations to compute $\mathcal{T}_{\mathcal{H}(p)}(C)$, where $P = \max\{p_A, p_B\}$ and $N = |\mathcal{I}| + |\mathcal{J}| + |\mathcal{K}|$.

Proof. Let $\mathbb{B} := \mathbb{B}_{\mathcal{I}\times\mathcal{J}}\cdot\mathbb{B}_{\mathcal{J}\times\mathcal{K}}$ denote the block tree and $\widetilde{p}_C := C_{\mathrm{sparse}}(\mathrm{depth}+1)\max\{p_A,p_B,C_{\mathrm{leaf}}\}$. According to Proposition 3.30, there holds $C \in \mathcal{H}(\mathbb{B}, \widetilde{p}_C)$.

1. step. For leaves $(\rho,\tau) \in \mathbb{B}_{\mathcal{I}\times\mathcal{K}}$ and $(\widetilde{\rho},\widetilde{\tau}) \in \mathbb{B}$ with $(\rho\times\tau)\cap(\widetilde{\rho}\times\widetilde{\tau}) \neq \emptyset$ holds either the inclusion $\rho\times\tau \subseteq \widetilde{\rho}\times\widetilde{\tau}$ or $\widetilde{\rho}\times\widetilde{\tau} \subseteq \rho\times\tau$: To see this, let $(i,k) \in (\rho\times\tau)\cap(\widetilde{\rho}\times\widetilde{\tau})$ so that $i \in \rho\times\widetilde{\rho}$ and $k \in \tau\times\widetilde{\tau}$. According to the definition of a cluster tree, this implies the inclusions

- $\rho \subseteq \widetilde{\rho}$ or $\widetilde{\rho} \subseteq \rho$

and

- $\tau \subseteq \widetilde{\tau}$ or $\widetilde{\tau} \subseteq \tau$.

Since $\operatorname{level}(\rho) = \operatorname{level}(\tau)$ and $\operatorname{level}(\widetilde{\rho}) = \operatorname{level}(\widetilde{\tau})$, this implies that

- either $\rho \subseteq \widetilde{\rho}$ and $\tau \subseteq \widetilde{\tau}$
- or $\widetilde{\rho} \subseteq \rho$ and $\widetilde{\tau} \subseteq \tau$.

2. step. For each leaf $(\rho, \tau) \in \mathbb{B}_{\mathcal{I}\times\mathcal{K}}$ holds $\operatorname{rank}(C|_{\rho\times\tau}) \leq p_C$: First, assume that there is a leaf $(\widetilde{\rho}, \widetilde{\tau}) \in \mathbb{B}$ with $\rho \times \tau \subseteq \widetilde{\rho} \times \widetilde{\tau}$. With $C_{\mathrm{id}} \geq 1$ and $p_C = C_{\mathrm{id}}\widetilde{p}_C$, we see

$$\operatorname{rank}(C|_{\rho\times\tau}) \leq \operatorname{rank}(C|_{\widetilde{\rho}\times\widetilde{\tau}}) \leq \widetilde{p}_C \leq p_C.$$

Otherwise, the leaf $\rho \times \tau$ is the finite union of leaves $\widetilde{\rho} \times \widetilde{\tau} \in \mathbb{B}$. Consequently,

$$\operatorname{rank}(C|_{\rho\times\tau}) \leq \left|\left\{(\widetilde{\rho}, \widetilde{\tau}) \in \mathbb{B} \text{ leaf} \,\middle|\, \widetilde{\rho} \times \widetilde{\tau} \subseteq \rho \times \tau\right\}\right| \widetilde{p}_C \leq C_{\mathrm{id}}\widetilde{p}_C = p_C.$$

3. step. Estimation of computational complexity: We first compute the representation $C = AB \in \mathcal{H}(\mathbb{B}, \widetilde{p}_C)$ with respect to the product tree in less than

$$3(\operatorname{depth}+1)^2 C_{\mathrm{sparse}}^3 \widehat{p}^{\,2}(|\mathcal{I}| + 2|\mathcal{J}| + |\mathcal{K}|)$$

operations, where $\widehat{p} := \max\{p_A, p_B, C_{\mathrm{leaf}}\}$. In a second step, we now transform the blocks of C into inadmissible or factorized rank-p blocks. Let $(\rho, \tau) \in \mathbb{P}^{(\mathcal{T}\times\mathcal{K})}$.

▶ We first consider the case $\rho \times \tau \subseteq \widetilde{\rho} \times \widetilde{\tau}$ for some $(\widetilde{\rho}, \widetilde{\tau}) \in \mathbb{B}$. In this case, there holds $C|_{\widetilde{\rho}\times\widetilde{\tau}} = \widetilde{U}\widetilde{V}^H$, and this provides a factorization $C|_{\rho\times\tau} = UV^H$ with $U \in \mathbb{K}^{|\rho|\times\widetilde{p}_C}$ and $V \in \mathbb{K}^{|\tau|\times\widetilde{p}_C}$.

- In case of $(\rho, \tau) \in \mathbb{P}_{\mathrm{near}}^{(\mathcal{I}\times\mathcal{K})}$, we have to transform $C|_{\rho\times\tau} = UV^H$ into a full matrix. This needs the complex conjugation of V and $|\rho|$ matrix-vector multiplications with U. The computational cost is thus less than

  $$|\tau|\widetilde{p}_C + |\tau||\rho|(2\widetilde{p}_C - 1) \leq 3C_{\mathrm{leaf}}\widetilde{p}_C\,(|\rho| + |\tau|) \leq 3\widetilde{p}_C^{\,2}\,(|\rho| + |\tau|)$$

  per near field block, since $\widetilde{p}_C \geq C_{\mathrm{leaf}}$.
- If $(\rho, \tau) \in \mathbb{P}_{\mathrm{far}}^{(\mathcal{I}\times\mathcal{K})}$, it needs less than

  $$6\widetilde{p}_C^{\,2}(|\rho| + |\tau|) + 23\widetilde{p}_C^{\,3} + |\rho|p$$

  operations to compute $\mathcal{T}_p(C|_{\rho\times\tau})$ in factorized form, cf. Corollary 3.8.

▶ It thus remains to consider the case when $\rho \times \tau$ is split into (at most) C_{id} blocks $\widetilde{\rho}_i \times \widetilde{\tau}_i$.

- In case of $(\rho, \tau) \in \mathbb{P}_{\mathrm{near}}^{(\mathcal{I}\times\mathcal{K})}$, we proceed as above and use matrix-vector multiplications on each block $(\widetilde{\rho}_i, \widetilde{\tau}_i)$. This leads to a computational complexity less than

  $$\sum_i 3\widetilde{p}_C^{\,2}\,(|\widetilde{\rho}_i| + |\widetilde{\tau}_i|) \leq 3C_{\mathrm{id}}p_C^2\,(|\rho| + |\tau|)$$

- In case of $(\rho,\tau) \in \mathbb{P}_{\text{far}}^{(\mathcal{I}\times\mathcal{K})}$, we proceed as for the coarsening of the block partitioning in Section 3.4.2 to obtain a factorization $C|_{\rho\times\tau} = UV^H$ out of the factorizations $C_{\widetilde{\rho}_i\times\widetilde{\tau}_i}$. This provides matrices $V \in \mathbb{K}^{|\rho|\times C_{\text{id}}\widetilde{p}_C^2}$ and $V \in \mathbb{K}^{|\tau|\times C_{\text{id}}\widetilde{p}_C^2}$. We therefore end up with a computational complexity of

$$6(C_{\text{id}}\widetilde{p}_C)^2(|\rho|+|\tau|) + 23(C_{\text{id}}\widetilde{p}_C)^3 + |\rho|p = 6C_{\text{id}}^2\widetilde{p}_C^2(|\rho|+|\tau|) + 23C_{\text{id}}^3\widetilde{p}_C^3 + |\rho|p$$

  for the computation of $\mathcal{T}_p(C|_{\rho\times\tau})$.

Consequently, the transformation of all nearfield and farfield blocks costs less than

$$\begin{aligned}\sum_{(\rho,\tau)\in\mathbb{P}^{(\mathcal{I}\times\mathcal{K})}} N_{\text{trans}}(C|_{\rho\times\tau}) &\leq 23C_{\text{id}}^3\widetilde{p}_C^3|\mathbb{P}_{\text{far}}^{(\mathcal{I}\times\mathcal{K})}| + (6C_{\text{id}}^2\widetilde{p}_C^2 + p)\sum_{(\rho,\tau)\in\mathbb{P}^{(\mathcal{I}\times\mathcal{K})}}(|\rho|+|\tau|)\\ &\leq 23C_{\text{id}}^3\widetilde{p}_C^3|\mathbb{P}_{\text{far}}^{(\mathcal{I}\times\mathcal{K})}| + 7C_{\text{sparse}}(\text{depth}+1)C_{\text{id}}^2\widetilde{p}_C^2(|\mathcal{I}|+|\mathcal{K}|)\\ &= C_{\text{sparse}}^3(\text{depth}+1)^3\big(23C_{\text{id}}^3\widehat{p}^3|\mathbb{P}_{\text{far}}^{(\mathcal{I}\times\mathcal{K})}| + 7C_{\text{id}}^2\widehat{p}^2(|\mathcal{I}|+|\mathcal{K}|)\big)\end{aligned}$$

Summing both contributions, we conclude the proof. ■

Remark. The computation of

$$\mathcal{T}_p\big((AB)|_{\rho\times\tau}\big) = \mathcal{T}_p\Big(\sum_{\ell=0}^{\text{level}(\rho\times\tau)}\sum_{\sigma\in\mathcal{U}_\ell(\rho\times\tau)} A|_{\rho\times\sigma}B|_{\sigma\times\tau}\Big)$$

can be rather expensive. One remedy is to truncate after each addition. This is called *fast truncation* in the literature. One can show that the complexity is then reduced from $C_{\text{id}}^3C_{\text{sparse}}^3(\text{depth}+1)^3p^3$ down to $C_{\text{id}}C_{\text{sparse}}^2p^2(p+1)^2 + C_{\text{id}}C_{\text{sparse}}(\text{depth}+1)p^3$. The complexity analysis and the precise definition of the fast truncation are found in Grasedyck-Hackbusch (2003). □

3.7 Further $\mathcal{H}$-Arithmetics (25.05.2009)

Based on the $\mathcal{H}$-addition and $\mathcal{H}$-multiplication, we can define further arithmetic operations, e.g., the $\mathcal{H}$-inversion or the $\mathcal{H}$-LU factorization. Contrary to the $\mathcal{H}$-addition and $\mathcal{H}$-multiplication, which are given (for constant local rank p) as best approximation operations, namely

$$A \oplus_p B := \mathcal{T}_{\mathcal{H}(p)}(A+B) \quad \text{and} \quad A \odot_p B := \mathcal{T}_{\mathcal{H}(p)}(AB),$$

the other $\mathcal{H}$-arithmetic operations won't, in general, compute the best approximation, e.g.,

$$\texttt{Inv}_{\mathcal{H}(p)}(A) \neq \mathcal{T}_{\mathcal{H}(p)}(A^{-1}).$$

However, one could compute all successive steps with adaptive $\mathcal{H}$-arithmetics instead to obtain arbitrarily accurate results.

3.7.1 $\mathcal{H}$-Inversion

One way to compute an approximate inverse of an $\mathcal{H}$-matrix has been proposed by GRASEDYCK (2001) and GRASEDYCK-HACKBUSCH (2003), and it is based on the Schur complement.

Lemma 3.33. *Assume that the $A \in \mathbb{K}^{n\times n}$ allows an LU factorization, i.e., all principal submatrices $(a_{jk})_{j,k=1,\dots,m}$ for all $1 \le m \le n$ are regular. We write A in block form*

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \tag{3.51}$$

with $A_{11} \in \mathbb{K}^{n_1\times n_1}$, $A_{12} \in \mathbb{K}^{n_1\times n_2}$, $A_{21} \in \mathbb{K}^{n_2\times n_1}$, $A_{22} \in \mathbb{K}^{n_2\times n_2}$, and $n = n_1 + n_2$. Then A_{11} as well as the **Schur complement**

$$S := A_{22} - A_{21}A_{11}^{-1}A_{12} \tag{3.52}$$

are invertible. Moreover, the inverse of A reads blockwise

$$A^{-1} = \begin{pmatrix} A_{11}^{-1} + A_{11}^{-1}A_{12}S^{-1}A_{21}A_{11}^{-1} & -A_{11}^{-1}A_{12}S^{-1} \\ -S^{-1}A_{21}A_{11}^{-1} & S^{-1} \end{pmatrix}. \tag{3.53}$$

Proof. Note that A_{11} is a principal submatrix and thus invertible by assumption. Since A is invertible as well, we use a block ansatz for the inverse $B := A^{-1}$:

$$\begin{pmatrix} \boldsymbol{I} & 0 \\ 0 & \boldsymbol{I} \end{pmatrix} \overset{!}{=} \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}$$

We consider the second column to see $0 = A_{11}B_{12} + A_{12}B_{22}$, whence $B_{12} = -A_{11}^{-1}A_{12}B_{22}$, and $\boldsymbol{I} = A_{21}B_{12} + A_{22}B_{22} = -A_{21}A_{11}^{-1}A_{12}B_{22} + A_{22}B_{22} = SB_{22}$. Therefore $SB_{22} = \boldsymbol{I}$ so that S has to be invertible. To see that (3.53) gives the inverse, one simply has to check that the product with A yields the identity $\mathbb{I}$. ■

Note that the explicit form (3.53) of the inverse gives rise to a recursive algorithm. If we replace all exact matrix additions and multiplications by $\mathcal{H}$-additions and $\mathcal{H}$-multiplications, respectively, we end up with an $\mathcal{H}$-inversion algorithm, which is stated explicitely in GRASEDYCK-HACKBUSCH (2003). The computational complexity of this algorithm is of the same size as the $\mathcal{H}$-matrix-matrix multiplication. Unfortunately, the recursive computation of A^{-1} by this Schur algorithm needs additional memory to store the A_{11}^{-1} blocks.

To make the latter algorithm meaningful, we note that $\mathcal{H}$-approximations usually satisfy the assumptions of Lemma 3.33 if the exact matrix, which has to be approximated, even does.

Lemma 3.34. *Let $A, B \in \mathbb{K}^{n\times n}$ and assume that A allows an LU factorization. Then, for sufficiently small error $\|A - B\|_2$, the matrix B allows an LU factorization as well.*

Proof. Recall that the determinant is a continuous polynomial, and invertible matrices $M \in \mathbb{K}^{m\times m}$ are characterized by $\det(M) \neq 0$. Therefore, the regular matrices

$$\left\{M \in \mathbb{K}^{m\times m} \,\middle|\, M \text{ regular}\right\} = \left\{M \in \mathbb{K}^{m\times m} \,\middle|\, \det(M) \neq 0\right\}$$

form an open subset of $\mathbb{K}^{m\times m}$. For each principal submatrix $A_m \in \mathbb{K}^{m\times m}$ of A, we may thus choose some $\varepsilon_m > 0$ such that the principal submatrix B_m is invertible provided that

$\|A_m - B_m\|_2 \leq \varepsilon_m$. Provided $\|A-B\|_2 \leq \min_{m=1,\dots,n} \varepsilon_m$, the estimate $\|A_m - B_m\|_2 \leq \|A-B\|_2$ thus proves that all principal submatrices of B are invertible. ■

3.7.2 Direct $\mathcal{H}$-Solvers

Before we state the $\mathcal{H}$-LU factorization, we have to consider triangular systems. We note that the idea of the following sections goes back to the doctoral thesis of LINTNER (2002), who introduced the $\mathcal{H}$-Cholesky factorization and a type of $\mathcal{H}$-QR factorization. The main contributions of the theses have been published in LINTNER (2004). BEBENDORF (2005) adopted these ideas for the computation of an $\mathcal{H}$-LU factorization. I could not find the precise complexity analysis as well as the error analysis in the literature. However, I am pretty sure that both can be achieved with the same techniques as before. It is, anyhow, clear that the error analysis will be pessimistic.

Linear Systems with Lower Triangular Matrix. Let $L \in \mathbb{K}^{n\times n}$ be a lower triangular matrix with $\ell_{jj} \neq 0$ and $B \in \mathbb{K}^{n\times m}$ a rectangular right-hand side. We want to compute the unique solution $X \in \mathbb{K}^{n\times m}$ of the linear system

$$LX = B. \tag{3.54}$$

To that end, we write this system in block form

$$\begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix}\begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix} = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}. \tag{3.55}$$

The recursive solution algorithm thus reads as follows:

- Compute X_{11} from $L_{11}X_{11} = B_{11}$.
- Compute X_{12} from $L_{11}X_{12} = B_{12}$.
- Compute X_{21} from $L_{22}X_{21} = B_{21} - L_{21}X_{11}$.
- Compute X_{22} from $L_{22}X_{22} = B_{22} - L_{21}X_{12}$.

Note that L_{11} as well as L_{22} satisfy the same assumptions as L so that we can guarantee the unique solvability in each step.

Remark. With the same strategy, one can solve $XU = B$ for $X \in \mathbb{K}^{m\times n}$, where $U \in \mathbb{K}^{n\times n}$ is an upper triangular matrix with $u_{jj} \neq 0$ and $B \in \mathbb{K}^{m\times n}$ is a given right-hand side. Theoretically, one simply considers the transposed system $U^T X^T = B^T$ and notices that $L := U^T$ is a lower triangular matrix. □

For $B \in \mathcal{H}(\mathbb{B}_{\mathcal{I}\times\mathcal{J}}, \mathbb{A}_{\mathcal{I}\times\mathcal{J}}, p)$, one usually stores X in the same format. Moreover, as can be seen from the stated algorithm, one can overwrite a block $B|_{\sigma\times\tau}$ by a corresponding block $X|_{\sigma\times\tau}$ of the solution matrix.

Linear Systems with Upper Triangular Matrix. The algorithm for the solution of lower triangular systems can simply be modified to the case $UX = B$, where $U \in \mathbb{K}^{n\times n}$ is an upper triangular matrix.

3.7.3 $\mathcal{H}$-LU Factorization

We assume that A allows an LU factorization. As before, we write A in block form and do a blockwise ansatz for the LU factorization

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix} \tag{3.56}$$

which leads to the following recursive algorithm:

- Compute L_{11} and U_{11} from $A_{11} = L_{11}U_{11}$.
- Compute U_{12} from $A_{12} = L_{11}U_{12}$.
- Compute L_{21} from $A_{21} = L_{21}U_{11}$.
- Compute L_{22} and U_{22} from $A_{22} = L_{21}U_{12} + L_{22}U_{22}$.

We note that the matrix blocks A_{jk} can be overwritten by the non-trivial blocks of L_{jk} and U_{jk}, respectively. If we replace the involved arithmetic operations by $\mathcal{H}$-arithmetics, we are led to an $\mathcal{H}$-algorithm which computes an approximate LU factorization.

3.7.4 $\mathcal{H}$-Inversion based on $\mathcal{H}$-LU Factorization

In this section, we provide a second algorithm to compute the $\mathcal{H}$-inverse which is due to Börm-Grasedyck-Hackbusch (2005). Compared to the prior algorithm based on the Schur complement, the advantage of the new algorithm is that it does not need any additional storage.

Assume that $A = LU$ is an LU factorization of the regular matrix $A \in \mathbb{K}^{n\times n}$. Then, $A^{-1} = U^{-1}L^{-1}$. We note that the regular upper triangular matrices as well as the regular lower triangular matrices form a group with respect to the (exact) matrix-matrix product. Therefore, U^{-1} is an upper triangular matrix and L^{-1} is a lower triangular matrix. We split the computation of A^{-1} into four steps

- Compute the LU factorization $A = LU$.
- Compute the inverse L^{-1} of L.
- Compute the inverse U^{-1} of U.
- Compute the matrix-matrix product $A^{-1} = U^{-1}L^{-1}$.

We stress that each of the steps can be done without further memory requirements, i.e. we can overwrite the matrix A.

Computation of L^{-1}. We assume that L is given in block form

$$L = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix}.$$

Obviously, the inverse of L then reads

$$L^{-1} = \begin{pmatrix} L_{11}^{-1} & 0 \\ -L_{22}^{-1}L_{21}L_{11}^{-1} & L_{22}^{-1} \end{pmatrix}. \tag{3.57}$$

The recursive computation is therefore performed as follows:

- Compute L_{11}^{-1}.
- Compute L_{22}^{-1}.
- Compute $-L_{22}^{-1}L_{21}L_{11}^{-1}$.

In any of the steps, we can overwrite the block of L to store the corresponding block of L^{-1}.

Computation of U^{-1}. We assume that U is given in block form

$$U = \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix}.$$

Obviously, the inverse of U then reads

$$U^{-1} = \begin{pmatrix} U_{11}^{-1} & -U_{22}^{-1}U_{12}U_{11}^{-1} \\ 0 & U_{22}^{-1} \end{pmatrix}. \tag{3.58}$$

The recursive computation is therefore performed as follows:

- Compute U_{11}^{-1}.
- Compute U_{22}^{-1}.
- Compute $-U_{22}^{-1}U_{12}L_{11}^{-1}$.

In any of the steps, we can overwrite the block of U to store the corresponding block of U^{-1}.

Computation of $A^{-1} = U^{-1}L^{-1}$. To keep notation simple, we write down the computation of UL, where U is an upper triangular and L is a lower triangular matrix. Written in block form, we obtain

$$\begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix}\begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} = \begin{pmatrix} U_{11}L_{11} + U_{12}L_{21} & U_{12}L_{22} \\ U_{22}L_{21} & U_{22}L_{22} \end{pmatrix}, \tag{3.59}$$

so that the computation is performed in the following steps:

- Compute $U_{11}L_{11} + U_{12}L_{21}$.
- Compute $U_{12}L_{22}$.
- Compute $U_{22}L_{21}$.
- Compute $U_{22}L_{22}$.

If it is done in this order, we may overwrite the blocks of A (which are usually already overwritten by the blocks of L and U, respectively).

3.7.5 $\mathcal{H}$-Cholesky Factorization

It is known from the basic lecture on numerical analysis that a matrix $A \in \mathbb{K}^{n\times n}$ can be factorized in the form $A = LL^H$ with some lower triangular matrix $L \in \mathbb{K}^{n\times n}$ with positive diagonal $\ell_{jj} > 0$, for all $j = 1, \ldots, n$, if and only if the matrix A is self-adjoint and positive definit (so-called SPD matrix). This factorization is called Cholesky factorization. Contrary to the LU factorization (and Gauss elimination), the use of the Cholesky provides a stable scheme to solve $Ax = b$ numerically.

For $\mathcal{H}$-matrices, a recursive algorithm to compute the $\mathcal{H}$-Cholesky factorization can be formulated along the lines of the $\mathcal{H}$-LU algorithm. Details are left to the reader.

Remark. To make the $\mathcal{H}$-Cholesky factorization meaningful, we have to ensure that the $\mathcal{H}$-matrix is SPD: As above, we assume that the exact matrix A, which is approximated by $\mathcal{H}$-matrices, is SPD. It is easy to guarantee that the $\mathcal{H}$-approximant A_p is selfadjoint as well, which has to be forced by the assembly algorithm. To see that A_p is positive definite, we note that the positive definiteness of A is equivalent to

$$\alpha := \inf_{\|x\|_2=1} Ax \cdot x > 0 \tag{3.60}$$

due to compactness (and the minimum is, in fact, obtained as the smallest eigenvalue of A). Moreover, there holds

$$\alpha_p := \inf_{\|x\|_2=1} A_p x \cdot x = \inf_{\|x\|_2=1} \big(Ax \cdot x - (A - A_p)x \cdot x \big) \geq \alpha - \|A - A_p\|_2. \tag{3.61}$$

Therefore, at least for sufficiently large p and small error $\|A - A_p\|_2$, the $\mathcal{H}$-approximant A_p is positive definite as well. □

3.7.6 $\mathcal{H}$-Inversion based on $\mathcal{H}$-Cholesky Factorization

As above in the case of the $\mathcal{H}$-LU factorization, the $\mathcal{H}$-inverse can be computed by use of the $\mathcal{H}$-Cholesky factorization. Details are left to the reader.

3.7.7 $\mathcal{H}$-QR Factorization

The main observation of Lintner (2002) was the following: Let $A \in \mathbb{K}^{m \times n}$ with $m \geq n$ and $\operatorname{rank}(A) = n$. Let $A = QR$ be the QR factorization of A. Plugging-in the QR factorization of A, we obtain $A^H A = R^H R$. Note that the QR factorization is unique provided there holds $r_{jj} > 0$ for the diagonal elements. With $L := R^H$, there holds $A^H A = LL^H$, i.e. the matrix $L \in \mathbb{K}^{n \times n}$ from the Cholesky factorization of the SPD matrix $A^H A$ determines the R-factor in the QR factorization of A.

Lintner therefore used the $\mathcal{H}$-Cholesky factorization to compute the matrix R of an approximate QR factorization. Knowing A and R, he then reconstructs an approximate orthogonal matrix Q. For details, the reader is referred to Lintner (2002). We stress that Lintner observes numerical instability (with respect to IEEE double precision) of his algorithm if the condition number of A is about 10^8. This is due to the condition number of $A^H A$ which is about 10^{16}, i.e. $A^H A$ is numerically singular.

3.7.8 Eigenvalue Computations in $\mathcal{H}$-Arithmetics

Besides the dissertation of Lintner (2002) and the corresponding article (2004), there are no publications on the approximate (and fast) computation of eigenvalues by means of $\mathcal{H}$-matrices. One idea could be to try the LU algorithm, which uses the same idea as the more prominent QR algorithm.

Chapter 4

Iterative Solution of Linear Systems

4.1 Krylov Spaces (04.06.2009)

Throughout the section, let $A \in \mathbb{K}^{n\times n}$ be a regular matrix and $b \in \mathbb{K}^n$ be a given right-hand side. We are concerned with the computation of the unique solution $x^* \in \mathbb{K}^n$ of $Ax^* = b$.

Definition. For $\ell \in \mathbb{N}$, we define the **Krylov spaces**

$$\mathcal{K}_\ell := \mathcal{K}_\ell(A, b) := \operatorname{span}\{b, Ab, A^2b, \ldots, A^{\ell-1}b\} \subseteq \mathbb{K}^n$$

Lemma 4.1. *For each $\ell \in \mathbb{N}$, the following is equivalent:*
(i) $\dim \mathcal{K}_{\ell+1} \leq \ell$,
(ii) $\mathcal{K}_\ell = \mathcal{K}_{\ell+1}$,
(iii) $A(\mathcal{K}_\ell) \subseteq \mathcal{K}_\ell$,
(iv) $x^* \in \mathcal{K}_\ell$.
In particular, there is an index $\ell \in \mathbb{N}$ such that $x^* \in \mathcal{K}_\ell$, and we define $\ell^* := \min\left\{\ell \in \mathbb{N} \,\middle|\, x^* \in \mathcal{K}_\ell\right\}$.

Proof. (i)⇒(ii): We choose the minimal $m < \ell$ such that the set $\{b, Ab, \ldots, A^m b\}$ is linearly dependent. According to the minimality of the index m, there are $\lambda_j \in \mathbb{R}$ with $A^m b = \sum_{j=0}^{m-1} \lambda_j A^j b$. In particular, there holds

$$A^\ell b = A^{\ell-m}(A^m b) = \sum_{j=0}^{m-1} \lambda_j A^{\ell-m+j} b \in \operatorname{span}\{A^{\ell-m}b, \ldots, A^{\ell-1}b\} \subseteq \mathcal{K}_\ell$$

and thus $\mathcal{K}_{\ell+1} \subseteq \mathcal{K}_\ell$.

(ii)⇒(iii): For $j = 0, \ldots, \ell-1$ holds $A(A^j b) \in \mathcal{K}_{\ell+1} = \mathcal{K}_\ell$. Since these vectors span the Krylov space $\mathcal{K}_\ell$, we are led to $A(\mathcal{K}_\ell) \subseteq \mathcal{K}_\ell$.

(iii)⇒(iv): Recall that A is a regular matrix which now additionally satisfies $A(\mathcal{K}_\ell) \subseteq \mathcal{K}_\ell$. Consequently, the linear mapping $A : \mathcal{K}_\ell \to \mathcal{K}_\ell$ is well-defined and injective, whence bijective. From $Ax^* = b \in \mathcal{K}_\ell$, we thus infer $x^* \in \mathcal{K}_\ell$.

(iv)⇒(i): There holds $x^* \in \mathcal{K}_\ell = \operatorname{span}\{b, Ab, A^2b, \ldots, A^{\ell-1}b\}$, whence $b = Ax^* \in \operatorname{span}\{Ab, A^2b, \ldots, A^\ell b\}$. In particular, the set $\{b, Ab, A^2b, \ldots, A^\ell b\}$ is linearly dependent, and we conclude $\dim \mathcal{K}_{\ell+1} \leq \ell$. ■

The following lemma is a standard result from Hilbert space theory and holds even in case of infinite dimensional spaces. However, we need the result only for finite dimensional Hilbert space, and the proof for finite dimensional spaces is elementary.

Lemma 4.2. *Let X be a finite dimensional Hilbert space and Y be a subspace of X. For any $x \in X$, there is a unique element $Px \in Y$ such that*

$$\|x - Px\|_X = \inf_{y \in Y} \|x - y\|_X. \tag{4.1}$$

The element $Px \in Y$ is called **best approximation of x in Y**. *The mapping $P : X \to Y$ is linear and satisfies*

$$Py = y \quad \text{and} \quad \langle x \,;\, y\rangle = \langle Px \,;\, y\rangle \quad \text{for all } y \in Y \text{ and } x \in X. \tag{4.2}$$

P is called **orthogonal projection onto Y**. *For a given orthonormal basis $y_1, \dots, y_n$ of Y, there holds*

$$Px = \sum_{j=1}^{n} \langle x \,;\, y_j\rangle y_j. \tag{4.3}$$

In particular, we have the **Parseval equality** $\|Px\|_X^2 = \sum_{j=1}^n |\langle x \,;\, y_j\rangle|^2$ *for all $x \in X$.*

Proof. We sketch the proof but leave the details to the reader:

1. step. From a compactness argument, we derive the solvability of (4.1).

2. step. An element $Px \in Y$ satisfies (4.1) if and only if there holds $\forall y \in Y \quad \langle x - Px \,;\, y\rangle = 0$.

3. step. The element $Px \in Y$ with (4.1) is unique and thus $P : X \to Y$ is well-defined.

4. step. The mapping $P : X \to Y$ is linear and satisfies (4.2).

5. step. If one writes Px as a linear combination of the orthonormal vectors y_j, one is led to (4.3) and proves, in particular, the Parseval equality. ∎

In the following, we state some of the so-called **Krylov methods** for the iterative solution of the linear system $Ax^* = b$. The most popular methods are the CG method and the GMRES method.

4.1.1 CG Method

Let $A \in \mathbb{K}^{n\times n}$ be an SPD matrix, i.e. A is selfadjoint $A = A^H$ with respect to the Euclidean scalar product $\langle x \,;\, y\rangle_2 := x \cdot y$ and positiv definite $\langle Ax \,;\, x\rangle_2 > 0$ for all $x \in \mathbb{K}^n \backslash \{0\}$. We then define the scalar product

$$\langle x \,;\, y\rangle_A := \langle x \,;\, Ay\rangle_2 \quad \text{for } x, y \in \mathbb{K}^n \tag{4.4}$$

The induced norm is $\|x\|_A = \langle x \,;\, x\rangle_A^{1/2}$. The **conjugate gradient method** means the computation of the sequence of best approximations $x_\ell \in \mathcal{K}_\ell(A, b)$ of x^*, for $\ell = 1, \dots, \ell^*$, with respect to the energy norm $\|\cdot\|_A$, i.e.,

$$\text{Find } x_\ell \in \mathcal{K}_\ell(A, b) \quad \text{such that} \quad \|x^* - x_\ell\|_A = \min_{y \in \mathcal{K}_\ell(A,b)} \|x^* - y\|_A. \tag{4.5}$$

Lemma 4.2 guarantees the unique existence of x_ℓ. Note that the computation of x_ℓ is possible without knowing x^*: To that end, we use the Gram-Schmidt orthogonalization algorithm to construct a sequence $d_0, \dots, d_{\ell-1}$ of pairwise $\langle \cdot \,;\, \cdot \rangle_A$-orthogonal vectors such that $\mathcal{K}_\ell(A,b) = \operatorname{span}\{d_0, \dots, d_{\ell-1}\}$. With (4.3), we then infer

$$x_\ell = \sum_{j=0}^{\ell-1} \frac{\langle x^* \,;\, d_j \rangle_A}{\langle d_j \,;\, d_j \rangle_A}\, d_j = \sum_{j=0}^{\ell-1} \frac{\langle b \,;\, d_j \rangle_2}{\langle d_j \,;\, d_j \rangle_A}\, d_j.$$

We stress that only b (but not x^*) is needed to compute x_ℓ.

4.1.2 CGNR Method

For a regular matrix $A \in \mathbb{K}^{n\times n}$, the **CG norm residual method** is the use of the CG method for the solution of the Gaussian equation $A^H A x^* = A^H b$. Note that

$$\|x^* - y\|_{A^H A}^2 = \langle x^* - y \,;\, A^H A(x^* - y) \rangle_2 = \langle A(x^* - y) \,;\, A(x^* - y) \rangle_2 = \|b - Ay\|_2^2,$$

whence the sequence of iterates x_ℓ satisfies

$$x_\ell \in \mathcal{K}_\ell(A^H A, A^H b) \quad \text{with} \quad \|b - Ax_\ell\|_2 = \min_{y \in \mathcal{K}_\ell(A^T A, A^T b)} \|b - Ay\|_2. \tag{4.6}$$

Again, the computation of x_ℓ is only based on the knowledge of b. The unique existence of x_ℓ follows from the CG method.

4.1.3 GMRES Method

For a regular matrix $A \in \mathbb{K}^{n\times n}$, the **generalized minimal residual method** is to compute a sequence of iterates x_ℓ such that

$$x_\ell \in \mathcal{K}_\ell(A,b) \quad \text{with} \quad \|b - Ax_\ell\|_2 = \min_{y \in \mathcal{K}_\ell(A,b)} \|b - Ay\|_2 \quad \text{for } \ell = 1, \dots, \ell^*. \tag{4.7}$$

We stress that this is (theoretically) a least square problem. Since the restriction of A to $\mathcal{K}_\ell(A,b)$ is injective, there is a unique solution x_ℓ of (4.7). Formally, the only difference to the CGNR method is that GMRES uses the Krylov spaces $\mathcal{K}_\ell(A,b)$, whereas CGNR uses the Krylov spaces $\mathcal{K}_\ell(A^H A, A^H b)$.

4.1.4 Remarks

Because of $x^* \in \mathcal{K}_{\ell^*}$, all of these Krylov methods compute in $\ell^* \le n$ steps the exact solution of $Ax^* = b$. Obviously, the efficient implementation is an important task. For the presentation, we restrict to the CG method which is the simplest method due to the strong assumptions on A. We shall see that each CG step needs only one matrix-vector multiplication. For the GMRES method, the reader is referred to, e.g., Plato [Section 11.6–11.8].

4.2 CG Method (04.06.2009)

Throughout, let $A \in \mathbb{K}^{n\times n}$ be an SPD matrix, $b \in \mathbb{K}^n\backslash\{0\}$ a given right-hand side and $\mathcal{K}_\ell := \mathcal{K}_\ell(A,b)$. Our first theorem essentially states that we can extend an orthogonal basis $\{d_0,\dots,d_{\ell-1}\}$ of $\mathcal{K}_\ell$ to an orthogonal basis $\{d_0,\dots,d_\ell\}$ of $\mathcal{K}_{\ell+1}$ by one matrix-vector multiplication.

Theorem 4.3. *We define the residuals*

$$r_0 := b \quad \text{and} \quad r_\ell := b - Ax_\ell \quad \text{for } \ell \in \mathbb{N},\ \ell = 1,\dots,\ell^*,$$

where $x_\ell \in \mathcal{K}_\ell$ are the CG iterates from (4.5), for $\ell \geq 1$, and $x_0 := 0$, respectively. For any $\ell < \ell^*$, $\{r_0,\dots,r_\ell\}$ is a basis of $\mathcal{K}_{\ell+1}$, and the Gram-Schmidt orthogonalization yields an $\langle\cdot\,;\,\cdot\rangle_A$-orthogonal basis $\{d_0,\dots,d_\ell\}$ of $\mathcal{K}_{\ell+1}$. Moreover, x_ℓ, r_ℓ, and d_ℓ can be computed inductively:
(i) *$d_0 = b$, $d_{\ell+1} = r_{\ell+1} + \beta_\ell d_\ell$ with $\beta_\ell := \|r_{\ell+1}\|_2^2/\|r_\ell\|_2^2$.*
(ii) *$x_{\ell+1} = x_\ell + \alpha_\ell d_\ell$, $r_{\ell+1} = r_\ell - \alpha_\ell A d_\ell$ with $\alpha_\ell := \|r_\ell\|_2^2/\|d_\ell\|_A^2$.*

Before we prove Theorem 4.3, we state the CG method algorithmically.

Algorithm 4.4 (CG method).

Input: SPD matrix $A \in \mathbb{K}^{n\times n}$, right-hand side $b \in \mathbb{K}^n$

Define $r_0 := b$, $d_0 := b$, $x_0 := 0$, $\ell := 0$.
(1) Stop, provided $r_\ell = 0$.
(2) Define $\alpha_\ell := \|r_\ell\|_2^2/\|d_\ell\|_A^2$, $\quad x_{\ell+1} := x_\ell + \alpha_\ell d_\ell$, $\quad r_{\ell+1} := r_\ell - \alpha_\ell A d_\ell$.
(3) Define $\beta_\ell := \|r_{\ell+1}\|_2^2/\|r_\ell\|_2^2$, $\quad d_{\ell+1} := r_{\ell+1} + \beta_\ell d_\ell$.
(4) Update $\ell \mapsto \ell+1$ and go to (1).

Output: Solution $x^* = x_\ell$ of $Ax^* = b$ as well as index $\ell^* = \ell$.

Remark. Due to numerical reasons, the stopping criterion in (1) has to be replaced by some appropriate criterion, e.g., $\|r_\ell\|_2 \leq \tau_{\text{abs}}$ with an absolute tolerance $\tau_{\text{abs}} > 0$. □

Remark. The stopping criterion for the CG method is essentially based on the norm of the residual. In many applications, a good approximation x_0 of x^* is known, e.g., from computations for a coarser discretization. So far, we have considered $x_0 = 0$. However, if x_0 is known, we modify the right hand side $\widetilde{b} := b - Ax_0$ and compute (an approximation of) the solution $\widetilde{x}^*$ of $A\widetilde{x}^* = \widetilde{b}$ by the CG method for $\mathcal{K}_\ell = \mathcal{K}_\ell(A,\widetilde{b})$. Then, there holds $x^* = \widetilde{x}^* + x_0$. □

Proof of Theorem 4.3. The proof is split into three major parts:

- First, we prove that $\{r_0,\dots,r_\ell\}$ is an ℓ_2-orthogonal basis of $\mathcal{K}_{\ell+1}$. Consequently, the Gram-Schmidt orthogonalization yields a $\langle\cdot\,;\,\cdot\rangle_A$-orthogonal basis $\{d_0,\dots,d_\ell\}$ of $\mathcal{K}_{\ell+1}$.
- Second, the vectors x_k, r_k, and d_k can be computed inductively with $\widetilde{\alpha}_k := \langle x^*\,;\,d_k\rangle_A/\|d_k\|_A^2$ and $\widetilde{\beta}_k := -\langle r_{k+1}\,;\,d_k\rangle_A/\|d_k\|_A^2$ replacing α_k and β_k, respectively.
- Third, there holds $\alpha_k = \widetilde{\alpha}_k$ and $\beta_k = \widetilde{\beta}_k$.

Each of the steps is proven by induction on ℓ separately, where the induction start $\ell = 0$ is obvious.

1. step. The residual r_k is ℓ_2-orthogonal to $\mathcal{K}_k$, i.e. $\langle r_k \,;\, y\rangle_2 = 0$ for all $y \in \mathcal{K}_k$: By definition, there holds $r_k \in \mathcal{K}_{k+1}$ and

$$\langle r_k \,;\, y\rangle_2 = \langle A(x^* - x_k) \,;\, y\rangle_2 = \langle x^* - x_k \,;\, y\rangle_A = 0 \quad \text{for all } y \in \mathcal{K}_k,$$

where we have used (4.2) for $Px^* = x_k$.

2. step. In particular, $\{r_0, \dots, r_\ell\} \subseteq \mathcal{K}_{\ell+1}$ is linearly independent and thus an ℓ_2-orthogonal basis of $\mathcal{K}_{\ell+1}$. Therefore, Gram-Schmidt orthogonalization of $\{r_0, \dots, r_\ell\}$ yields an $\langle\cdot\,;\,\cdot\rangle_A$-orthogonal basis $\{d_0, \dots, d_\ell\}$ of $\mathcal{K}_{\ell+1}$ with $d_0 = r_0 = b$.

3. step. With $\widetilde{\alpha}_k := \langle x^* \,;\, d_k\rangle_A / \|d_k\|_A^2$, there holds $x_{k+1} = x_k + \widetilde{\alpha}_k d_k$ and $r_{k+1} = r_k - \widetilde{\alpha}_k A d_k$: With the orthogonal basis $\{d_0, \dots, d_k\}$ of $\mathcal{K}_{k+1}$, (4.3) proves

$$x_{k+1} = \sum_{j=0}^{k} \frac{\langle x^* \,;\, d_j\rangle_A}{\|d_j\|_A^2}\, d_j = x_k + \frac{\langle x^* \,;\, d_k\rangle_A}{\|d_k\|_A^2}\, d_k = x_k + \widetilde{\alpha}_k d_k.$$

By definition of $r_{k+1} = b - Ax_{k+1}$, this implies $r_{k+1} = r_k - \widetilde{\alpha}_k A d_k$.

4. step. With $\widetilde{\beta}_k := -\langle r_{k+1} \,;\, d_k\rangle_A / \|d_k\|_A^2$, there holds $d_{k+1} = r_{k+1} + \widetilde{\beta}_k d_k$: According to the definition of the Gram-Schmidt orthogonalization, there holds

$$d_{k+1} = r_{k+1} - \sum_{j=0}^{k} \frac{\langle r_{k+1} \,;\, d_j\rangle_A}{\|d_j\|_A^2}\, d_j$$

From $r_j \in \mathcal{K}_{j+1}$ we obtain inductively $d_j \in \mathcal{K}_{j+1}$. For $j \le k-1$, it follows that $d_j \in \mathcal{K}_k$, whence $Ad_j \in \mathcal{K}_{k+1}$. Therefore, step 1 implies

$$\langle r_{k+1} \,;\, d_j\rangle_A = \langle r_{k+1} \,;\, Ad_j\rangle_2 = 0 \quad \text{for } j \le k-1,$$

i.e. the sum is reduced to the last summand for $j = k$.

5. step. There holds $\widetilde{\alpha}_k := \langle x^* \,;\, d_k\rangle_A / \|d_k\|_A^2 = \|r_k\|_2^2 / \|d_k\|_A^2 =: \alpha_k$: According to step 3, there holds $d_k = r_k + \widetilde{\beta}_{k-1} d_{k-1}$. With step 1 and $d_{k-1} \in \mathcal{K}_k$, we infer $\langle r_k \,;\, d_{k-1}\rangle_2 = 0$. Therefore,

$$\|r_k\|_2^2 = \langle r_k \,;\, r_k\rangle_2 = \langle r_k \,;\, d_k\rangle_2 = \langle b - Ax_k \,;\, d_k\rangle_2 = \langle x^* - x_k \,;\, d_k\rangle_A = \langle x^* \,;\, d_k\rangle_A,$$

since $x_k \in \mathcal{K}_k$ and thus $\langle x_k \,;\, d_k\rangle_A = 0$.

6. step. There holds $\widetilde{\beta}_k := -\langle r_{k+1} \,;\, d_k\rangle_A / \|d_k\|_A^2 = \|r_{k+1}\|_2^2 / \|r_k\|_2^2 =: \beta_k$. With step 3 and step 5 holds $r_{k+1} = r_k - \alpha_k A d_k$. From $\langle r_{k+1} \,;\, r_k\rangle_2 = 0$, we obtain

$$\|r_{k+1}\|_2^2 = -\alpha_k \langle r_{k+1} \,;\, Ad_k\rangle_2 = -\alpha_k \langle r_{k+1} \,;\, d_k\rangle_A = \widetilde{\beta}_k \|r_k\|_2^2$$

according to the definition of $\widetilde{\beta}_k$ and α_k. ■

One major advantage of the CG method is that one can estimate the number of iterations to obtain an approximation x_ℓ such that the error $\|x^* - x_\ell\|_2$ is sufficiently small. In particular, the following a priori error estimate states that the error essentially depends on the condition number of A.

Theorem 4.5. *With $\kappa := \sqrt{\mathrm{cond}_2(A)}$ there holds, for all $\ell \in \mathbb{N}$, the relative error estimate*

$$\kappa^{-1} \frac{\|x^* - x_\ell\|_2}{\|x^*\|_2} \le \frac{\|x^* - x_\ell\|_A}{\|x^*\|_A} \le 2\Big(\frac{\kappa - 1}{\kappa + 1}\Big)^\ell \tag{4.8}$$

Proof. Let $\lambda_1 \ge \cdots \ge \lambda_n > 0$ be the eigenvalues of A. Note that $\mathrm{cond}_2(A) = \lambda_1/\lambda_n$. We may choose an ℓ_2-orthonormal basis $\{v_1, \dots, v_n\}$ of $\mathbb{K}^n$ such that v_j is eigenvector of A for the eigenvalue λ_j. For any vector $x = \sum_{j=1}^n \alpha_j v_j \in \mathbb{K}^n$ holds

$$\|x\|_2^2 = \sum_{j=1}^n |\alpha_j|^2 \quad \text{and} \quad \|x\|_A^2 = \langle Ax\,;\, x\rangle_2 = \sum_{j=1}^n \lambda_j |\alpha_j|^2 \tag{4.9}$$

We therefore obtain the norm equivalence

$$\sqrt{\lambda_n}\,\|x\|_2 \le \|x\|_A \le \sqrt{\lambda_1}\,\|x\|_2 \quad \text{for } x \in \mathbb{K}^d,$$

whence the first estimate in (4.8)

$$\frac{\|x^* - x\|_2}{\|x^*\|_2} \le \frac{\sqrt{\lambda_1}}{\sqrt{\lambda_n}} \frac{\|x^* - x\|_A}{\|x^*\|_A} = \kappa\, \frac{\|x^* - x\|_A}{\|x^*\|_A}.$$

In case of $\mathrm{cond}_2(A) = \lambda_1/\lambda_n = 1$, there holds $A = \lambda_1 \boldsymbol{I}$ and whence $x_1 = x^*$. Without loss of generality, we may therefore assume $\kappa > 1$. The second estimate in (4.8) is proven within two steps.

1. step. There holds the estimate

$$\frac{\|x^* - x_\ell\|_A}{\|x^*\|_A} \le \inf_{p \in \mathbb{P}_\ell, p(0)=1} \max_{j=1,\dots,n} |p(\lambda_j)| :$$

Let $p \in \mathbb{P}_\ell$ with $p(0) = 1$. By polynomial division, there is a polynomial $q \in \mathbb{P}_{\ell-1}$ such that $tq(t) = 1 - p(t)$, whence $p(t) = 1 - tq(t)$. Note that therefore

$$p(A)x^* = \big(1 - Aq(A)\big)x^* = x^* - q(A)b \quad \text{and} \quad q(A)b \in \mathcal{K}_\ell.$$

Writing $x^* = \sum_{j=1}^n \alpha_j v_j$, we obtain

$$\begin{aligned}\|x^* - x_\ell\|_A^2 \le \|x^* - q(A)b\|_A^2 = \|p(A)x^*\|_A^2 = \sum_{j=1}^n \lambda_j \alpha_j^2 p(\lambda_j)^2 &\le \max_{k=1,\dots,\ell} |p(\lambda_k)|^2 \sum_{j=1}^n \lambda_j \alpha_j^2 \\ &= \max_{k=1,\dots,\ell} |p(\lambda_k)|^2\, \|x^*\|_A^2.\end{aligned}$$

2. step. There is a polynomial $p \in \mathbb{P}_\ell$ with $p(0) = 1$ and

$$|p(\lambda_j)| \le 2\Big(\frac{\kappa - 1}{\kappa + 1}\Big)^\ell \quad \text{for all } j = 1, \dots, n :$$

With the Čebyšev polynomial $T_\ell(t) := \arccos(\ell \cos(t)) \in \mathbb{P}_\ell$ on $[-1, 1]$, we define

$$p(\lambda) := \frac{q(\lambda)}{q(0)}, \quad q(\lambda) := T_\ell\Big(\frac{\lambda_1 + \lambda_n - 2\lambda}{\lambda_1 - \lambda_n}\Big)$$

Obviously, there holds $p \in \mathbb{P}_\ell$ with $p(0) = 1$. Moreover, from $\|T_\ell\|_{\infty,[-1,1]} = 1$ and $\kappa = \lambda_1/\lambda_n$, we obtain

$$\max_{\lambda_n \le \lambda \le \lambda_1} |p(\lambda)| \le \frac{1}{|q(0)|} = \Big|T_\ell\Big(\frac{\lambda_1 + \lambda_n}{\lambda_1 - \lambda_n}\Big)\Big|^{-1} = \Big|T_\ell\Big(\frac{\mathrm{cond}_2(A) + 1}{\mathrm{cond}_2(A) - 1}\Big)\Big|^{-1}.$$

Finally, a laborious direct calculation for the Čebyšev polynomial T_ℓ proves

$$\Big|T_\ell\Big(\frac{\kappa^2 + 1}{\kappa^2 - 1}\Big)\Big| \ge \frac{1}{2}\left(\frac{\kappa + 1}{\kappa - 1}\right)^\ell,$$

c.f. Plato [Lemma 11.18]. This concludes the proof. ■

Symmetric Preconditioning

As we have seen, the decay of the relative error $\|x^* - x_\ell\|_2/\|x^*\|_2$ essentially depends on the condition number of the SPD matrix A. The larger $\mathrm{cond}_2(A)$, the worse is the convergence of the CG method. Assume that we can compute the Cholesky decomposition $P = LL^H$ of an SPD matrix $P \approx A$. Since the Cholesky decomposition depends continuously on P, there holds

$$\widetilde{A} := L^{-1}AL^{-H} \approx \boldsymbol{I},$$

i.e. $1 \approx \mathrm{cond}_2(\widetilde{A}) \ll \mathrm{cond}_2(A)$. Note that $\widetilde{A}$ is an SPD matrix, too, since

$$\widetilde{A} = \widetilde{A}^H \quad \text{and} \quad \widetilde{A}x \cdot x = (AL^{-H}x) \cdot (L^{-H}x) > 0 \quad \text{for } x \neq 0.$$

With $\widetilde{b} := L^{-1}b$, we may therefore use the CG method with respect to $\mathcal{K}_\ell = \mathcal{K}_\ell(\widetilde{A}, \widetilde{b})$ to compute the solution $\widetilde{x}^*$ of $\widetilde{A}\widetilde{x}^* = \widetilde{b}$. Finally, $x^* := L^{-H}\widetilde{x}^*$ solves $L^{-1}Ax^* = L^{-1}AL^{-H}\widetilde{x}^* = \widetilde{b} = L^{-1}b$, whence $Ax^* = b$.

Exercise. Formulate the CG algorithm with symmetric preconditioning. Avoid to compute L^{-1} and L^{-H} but solve appropriate linear systems instead. For instance, the vector $y := L^{-H}x$ can be obtained by solving the linear system $L^H y = x$, and L^H is an upper triangular matrix. □

Non-Symmetric Preconditioning

Suppose that $A, P \in \mathbb{K}^{n\times n}$ are SPD matrices with $P \approx A$. Let the Cholesky factorization $P = LL^H$ be given. From a conceptual point of view, it would be convenient to consider the matrix $P^{-1}A$ in the CG method. Unfortunately, there holds $(P^{-1}A)^H = A^H P^{-H} = AP^{-1}$, i.e. the matrix $P^{-1}A$ is *not* selfadjoint — with respect to the Euclidean scalar product $\langle x\,;\,y\rangle_2 := x \cdot y$.

Exercise. Suppose that $A, P \in \mathbb{K}^{n\times n}$ are SPD matrices with $P \approx A$ and define a scalar product $\langle x\,;\,y\rangle_P := (Px) \cdot y$. Prove that $P^{-1}A$ is an SPD matrix with respect to $\langle\cdot\,;\,\cdot\rangle_P$. How, do Theorem 4.3 and Theorem 4.5 look like if we replace the Euclidean scalar product $\langle x\,;\,y\rangle_2 := x \cdot y$ by $\langle\cdot\,;\,\cdot\rangle_P$ and if we apply the CG method for the matrix $P^{-1}A$ with respect to $\langle\cdot\,;\,\cdot\rangle_P$? How does Algorithm 4.4 look like and how would you realize the matrix-vector multiplication with the matrix $P^{-1}A$? □

Chapter 5

$\mathcal{H}^2$-Matrices

5.1 $\mathcal{H}^2$-Matrices by Interpolation (18.06.2009)

We consider again the case of a dense matrix $A \in \mathbb{K}^{m\times n}$ whose entries have a special structure, namely

$$A_{ij} = \kappa(x_i, y_j) \quad \text{for all } i = 1,\dots,m \text{ and } j = 1,\dots,n \tag{A1}$$

with given evaluation points $x_i, y_j \in \mathbb{R}^d$ or

$$A_{ij} = \int_\Omega \kappa(x_i, y)\psi_j(y)\,dy \quad \text{for all } i = 1,\dots,m \text{ and } j = 1,\dots,n \tag{A2}$$

with given evaluation points $x_i \in \mathbb{R}^d$ and functions $\psi_j \in L^1(\Omega)$ or

$$A_{ij} = \int_\omega \int_\Omega \phi_i(x)\kappa(x, y)\psi_j(y)\,dy\,dx \quad \text{for all } i = 1,\dots,m \text{ and } j = 1,\dots,n \tag{A3}$$

with given functions $\phi_i \in L^1(\omega)$ and $\psi_j \in L^1(\Omega)$. Here, $\kappa(x, y)$ is an asymptotically smooth kernel. Let $\mathbb{T}_\mathcal{I}$ and $\mathbb{T}_\mathcal{J}$ be cluster trees for $\mathcal{I} = \{1,\dots,m\}$ and $\mathcal{J} = \{1,\dots,n\}$, respectively.

Modified Interpolation Approach. We consider the strengthened admissibility condition

$$\operatorname{diam}(B_\sigma \times B_\tau) \le \eta \operatorname{dist}(B_\sigma, B_\tau) \tag{$\mathbb{A}$}$$

which corresponds to replace the minimum in ($\mathbb{A}$) by a maximum. We then build the block partitioning $\mathbb{B}_{\mathcal{I}\times\mathcal{J}} = \mathbb{B}(\mathbb{T}_\mathcal{I}, \mathbb{T}_\mathcal{J}, \mathbb{A})$ with respect to this new admissibility condition. Now, we assume that $(\sigma, \tau) \in \mathbb{P}_{\text{far}}$. We then replace κ on $B := B_\sigma \times B_\tau$ by simultaneous interpolation with respect to the x- and y-variable, i.e.

$$\widetilde{\kappa}_p^{\sigma\tau}(x, y) := \mathcal{I}_p^{\sigma\tau}\kappa(x, y) = \sum_{k,\ell=1}^{p^d} \kappa(x_k^\sigma, y_\ell^\tau)\mathcal{L}_k^\sigma(x)\mathcal{L}_\ell^\tau(y) \tag{5.1}$$

with tensorial interpolation nodes $x_k^\sigma \in B_\sigma$ and $y_\ell^\tau \in B_\tau$, respectively. For instance, let us consider (A3). For $i \in \sigma$ and $j \in \tau$, there holds

$$A_{ij} \approx \sum_{k,\ell=1}^{p^d} \kappa(x_k^\sigma, y_\ell^\tau)\Big(\int_\omega \phi_i(x)\mathcal{L}_k^\sigma(x)\,dx\Big)\Big(\int_\Omega \psi_j(y)\mathcal{L}_\ell^\tau(y)\,dy\Big) =: (A_p)_{ij}. \tag{5.2}$$

If we define the cluster basis matrices $V^\sigma \in \mathbb{K}^{|\sigma|\times p^d}$ and $W^\tau \in \mathbb{K}^{|\tau|\times p^d}$ by

$$V^\sigma_{ik} := \int_\omega \phi_i(x)\mathcal{L}^\sigma_k(x)\,dx \qquad \text{and} \qquad V^\tau_{j\ell} := \int_\Omega \psi_j(y)\mathcal{L}^\tau_\ell(y)\,dy$$

as well as the multiplication matrix $M^{\sigma\tau} \in \mathbb{K}^{p^d\times p^d}$ by

$$M^{\sigma\tau}_{k\ell} := \kappa(x^\sigma_k, y^\tau_\ell),$$

Equation (5.2) becomes

$$A|_{\sigma\times\tau} \approx V^\sigma M^{\sigma\tau}(W^\tau)^T =: A_p|_{\sigma\times\tau}. \tag{5.3}$$

In particular, we observe that $\operatorname{rank}(A_p|_{\sigma\times\tau}) \le p^d$ and thus $A_p \in \mathcal{H}(\mathbb{B}_{\mathcal{I}\times\mathcal{J}}, \mathbb{A}, p^d)$ with the block partition $\mathbb{B}_{\mathcal{I}\times\mathcal{J}}$ built from $\mathbb{T}_\mathcal{I}$ and $\mathbb{T}_\mathcal{J}$.

Remark. As one side result of this interpolation approach, we observe that we don't have to perform an integration of the kernel κ (with respect to either the x- or y-variable). Since the ansatz functions are usually piecewise polynomials, the arising integrals can now be computed analytically by use of (e.g. Gaussian) quadrature rules. In particular, we thus avoid any quadrature error when compared to the interpolation approach of Section 2. □

The Second Hierarchy. Another benefit of this new approach is that it leads to a second hierarchy (besides the hierarchy of the cluster tree) for the cluster bases $(V^\sigma)_{\sigma\in\mathbb{T}_\mathcal{I}}$ and $(W^\tau)_{\tau\in\mathbb{T}_\mathcal{J}}$. This additional structure will allow to reduce the computational cost of the matrix-vector multiplication from generically $\mathcal{O}(n\log n)$ down to $\mathcal{O}(n)$. To observe the second hierarchy, recall that polynomial interpolation is a projection: For $\sigma' \in \operatorname{sons}(\sigma)$, we have

$$\mathcal{L}^\sigma_k(x) = \sum_{\ell=1}^{p^d} \mathcal{L}^\sigma_k(x^{\sigma'}_\ell)\mathcal{L}^{\sigma'}_\ell(x). \tag{5.4}$$

Plugging this into the definition of V^σ_{ik}, we prove for $i \in \sigma' \in \operatorname{sons}(\sigma)$

$$V^\sigma_{ik} = \int_\omega \phi_i\mathcal{L}^\sigma_k\,dx = \sum_{\ell=1}^{p^d} \mathcal{L}^\sigma_k(x^{\sigma'}_\ell)V^{\sigma'}_{ik}.$$

Which can be written blockwise by

$$V^\sigma|_{\sigma'} = V^{\sigma'}T^{\sigma'\sigma}, \tag{5.5}$$

where the transfer matrix $T^{\sigma'\sigma} \in \mathbb{K}^{p^d\times p^d}$ is defined by

$$T^{\sigma'\sigma}_{\ell k} := \mathcal{L}^\sigma_k(x^{\sigma'}_\ell). \tag{5.6}$$

Thus, we have observed an additional relation between the cluster basis matrices V^σ of father and son: If we know all transfer matrices $T^{\sigma'\sigma}$, we only need to store the matrices V^σ for the leaves of $\mathbb{T}$. This is the second hierarchy leading to $\mathcal{H}^2$-matrices. Note that $(W^\tau)_{\tau\in\mathbb{T}_\mathcal{J}}$ has the same property, namely $W^\tau|_{\tau'} = W^{\tau'}T^{\tau'\tau}$ for all $\tau \in \mathbb{T}_\mathcal{J}$ and $\tau' \in \operatorname{sons}(\tau)$.

A Priori Error Control. We stress that Proposition 2.9 also applies for this new interpolation ansatz. To prove this, we only have to provide the analogon to Theorem 2.13 for simultaneous interpolation in x- and y-direction.

Theorem 5.1. *Let κ be an asymptotically smooth kernel and B_σ, B_τ non-degenerate boxes in $\mathbb{R}^d$ with $\operatorname{diam}(B_\sigma \times B_\tau) \leq \eta \operatorname{dist}(B_\sigma, B_\tau)$ for some $\eta > 0$. If $\mathcal{I}_p^{\sigma\tau}$ denotes the tensorial Čebyšev interpolation in x- and y-direction with interpolation nodes in $B_\sigma \times B_\tau$, there holds*

$$\|\kappa - \mathcal{I}_p^{\sigma\tau}\kappa\|_{\infty, B_\sigma \times B_\tau} \leq 8dc_1(\eta/c_2)^s \operatorname{diam}(B_\tau)^{-s} \Lambda_p^{2d-1} \left(\frac{\eta}{4c_2}\right)^p \tag{5.7}$$

i.e. we obtain exponential convergence with respect to p provided $\eta < 4c_2$ since Λ_p grows only logarithmically.

Proof. We apply Corollary 2.12 on the box $B := B_\sigma \times B_\tau \subset \mathbb{R}^{2d}$. Thus, one only has to replace the dimension d in the proof of Theorem 2.13 by the dimension $2d$. ■

5.2 Complexity Analysis for $\mathcal{H}^2$-Matrices (18.06.2009)

The following definition is a slight generalization from the previous section.

Definition. Let $\mathbb{B}_{\mathcal{I}\times\mathcal{J}}$ be a block partitioning introduced by cluster trees $\mathbb{T}_\mathcal{I}$ and $\mathbb{T}_\mathcal{J}$ and an appropriate admissibility condition $\mathbb{A}$. A matrix $A \in \mathbb{K}^{m\times n}$ is called **uniform $\mathcal{H}$-matrix**, with respect to $\mathbb{B}_{\mathcal{I}\times\mathcal{J}}$, if there holds the following: There are families $(V_\sigma)_{\sigma\in\mathbb{T}_\mathcal{I}}$ and $(W_\tau)_{\tau\in\mathbb{T}_\mathcal{J}}$ of matrices $V_\sigma \in \mathbb{K}^{|\sigma|\times p}$ and $W_\tau \in \mathbb{K}^{|\tau|\times p}$ called **cluster bases** and a family $(M_{\sigma\tau})_{(\sigma,\tau)\in\mathbb{P}_{\text{far}}}$ of **multiplication matrices** with

$$A|_{\sigma\times\tau} = V_\sigma M_{\sigma\tau} W_\tau^H \quad \text{for all } (\sigma,\tau) \in \mathbb{P}_{\text{far}}. \tag{5.8}$$

Moreover, a uniform $\mathcal{H}$-matrix is called **$\mathcal{H}^2$-matrix** if there are families of **transfer matrices** $S_{\sigma'\sigma} \in \mathbb{K}^{p\times p}$ and $T_{\tau'\tau} \in \mathbb{K}^{p\times p}$ such that

$$V_\sigma|_{\sigma'} = V_{\sigma'} S_{\sigma'\sigma} \quad \text{for all } \sigma \in \mathbb{T}_\mathcal{I}, \sigma' \in \operatorname{sons}(\sigma) \tag{5.9}$$

as well as

$$W_\tau|_{\tau'} = W_{\tau'} T_{\tau'\tau} \quad \text{for all } \tau \in \mathbb{T}_\mathcal{J}, \tau' \in \operatorname{sons}(\tau) \tag{5.10}$$

For this additional assumption, one says that the cluster bases (V_σ) resp. (W_τ) are **nested**. If $A \in \mathbb{K}^{m\times n}$ is an $\mathcal{H}^2$-matrix, we shall write $A \in \mathcal{H}^2(\mathbb{B}_{\mathcal{I}\times\mathcal{J}}, \mathbb{A}, p)$.

Obviously, any uniform $\mathcal{H}$-matrix or $\mathcal{H}^2$-matrix $A \in \mathbb{K}^{m\times n}$ is a special $\mathcal{H}$-matrix. To gain the linear complexity with respect to the problem size $N = m + n$, one has to guarantee, in particular, that the near field has only linear storage requirements. In general this is not true for the block partitioning given by Algorithm 2.3. Therefore, we modify the algorithm slightly:

Algorithm 5.2 (Create Block Partitioning (Second Version)).

```
function CreateBlockPartitioning(ℙ_near,ℙ_far,σ,τ)
if (σ,τ) admissible
   add (σ,τ) to ℙ_far
elseif sons(σ) ≠ ∅
   if sons(τ) ≠ ∅
      for all (σ',τ') ∈ sons(σ) × sons(τ)
         call CreateBlockPartitioning(ℙ_near,ℙ_far,σ',τ')
   else
      for all σ' ∈ sons(σ)
         call CreateBlockPartitioning(ℙ_near,ℙ_far,σ',τ)
   end
elseif sons(τ) ≠ ∅
   for all τ' ∈ sons(τ)
      call CreateBlockPartitioning(ℙ_near,ℙ_far,σ,τ')
else /* inadmissible block which cannot be split any further */
   add (σ,τ) to ℙ_near
end
```

Remark. For an $\mathcal{H}^2$-matrix A one only stores the near field $A|_{\sigma\times\tau}$ for $(\sigma,\tau)\in\mathbb{P}_{\text{near}}$, the multiplication matrices $M_{\sigma\tau}$ for the far field $(\sigma,\tau)\in\mathbb{P}_{\text{far}}$, the transfer matrices $S_{\sigma'\sigma}$ and $T_{\tau'\tau}$ for any father-son pair (σ,σ') and (τ,τ'), and the cluster basis matrices V_σ and W_τ for all leaves $\sigma\in\mathbb{T}_{\mathcal{I}}$ and $\tau\in\mathbb{T}_{\mathcal{J}}$ (i.e. *not* for all clusters). □

5.2.1 Storage Requirements for $\mathcal{H}^2$-Matrices

Proposition 5.3. *Assume that the block partitioning $\mathbb{P}$ has been created by Algorithm 5.2. Then the storage requirements N_{Storage} for an $\mathcal{H}^2$-matrix A satisfy*

$$N_{\text{Storage}} \leq \left(|\mathbb{T}_{\mathcal{I}}| + |\mathbb{T}_{\mathcal{J}}|\right)\left(p^2 + \frac{1}{2}C_{\text{sparse}}\max\{C_{\text{leaf}}^2, p^2\}\right) + (m+n)p = \mathcal{O}(p^2 N),$$

where $N = m + n$.

Proof. The number of blocks in $\mathbb{P}$ satisfies $|\mathbb{P}| \leq C_{\text{sparse}}\min\{|\mathbb{T}_{\mathcal{I}}|, |\mathbb{T}_{\mathcal{J}}|\}$. A near field block (σ,τ) needs at most C_{leaf}^2 memory units, a multiplication matrix $M_{\sigma\tau}$ for (σ,τ) needs p^2 memory units. Thus, we need less than

$$\begin{aligned}|\mathbb{P}|\,\max\{p^2, C_{\text{leaf}}^2\} &\leq C_{\text{sparse}}\min\{|\mathbb{T}_{\mathcal{I}}|, |\mathbb{T}_{\mathcal{J}}|\}\,\max\{p^2, C_{\text{leaf}}^2\}\\ &\leq \frac{1}{2}C_{\text{sparse}}(|\mathbb{T}_{\mathcal{I}}| + |\mathbb{T}_{\mathcal{J}}|)\max\{p^2, C_{\text{leaf}}^2\}\end{aligned}$$

memory units to store all near field blocks and all multiplication matrices. The storage of the transfer matrices costs less than $p^2(|\mathbb{T}_{\mathcal{I}}| + |\mathbb{T}_{\mathcal{J}}|)$, since one has less than $|\mathbb{T}|$ father-son pairings per cluster tree. Finally, the matrices V_σ and W_τ have only to be stored for the leaves of cluster

trees $\mathbb{T}_{\mathcal{I}}$ and $\mathbb{T}_{\mathcal{J}}$: This leads to an additional storage requirement of

$$\sum_{\substack{\sigma\in\mathbb{T}_{\mathcal{I}}\\ \sigma \text{ leaf}}} |\sigma|p = mp \quad \text{and} \quad \sum_{\substack{\tau\in\mathbb{T}_{\mathcal{J}}\\ \tau \text{ leaf}}} |\tau|p = np$$

memory units. ■

5.2.2 Matrix-Vector Multiplication with $\mathcal{H}^2$-Matrices

As for common $\mathcal{H}$-matrices, we write the matrix-vector multiplication blockwise,

$$(Ax)_i = \sum_{\substack{(\sigma,\tau)\in\mathbb{P}_{\text{near}}\\ i\in\sigma}} (A|_{\sigma\times\tau}x|_\tau)_i + \sum_{\substack{(\sigma,\tau)\in\mathbb{P}_{\text{far}}\\ i\in\sigma}} (V_\sigma M_{\sigma\tau} W_\tau^H x|_\tau)_i$$

with $1 \le i \le m$. We define families $(\widehat{x}_\tau)_{\tau\in\mathbb{T}_{\mathcal{J}}}$ and $(\widehat{y}_\sigma)_{\sigma\in\mathbb{T}_{\mathcal{I}}}$ of vectors $\widehat{x}_\tau, \widehat{y}_\sigma \in \mathbb{K}^p$ as follows:

$$\begin{aligned} \widehat{x}_\tau &:= W_\tau^H x|_\tau, && \text{for all } \tau \in \mathbb{T}_{\mathcal{J}},\\ \widehat{y}_\sigma &:= \sum_{\substack{\tau\in\mathbb{T}_{\mathcal{J}}\\ (\sigma,\tau)\in\mathbb{P}_{\text{far}}}} M_{\sigma\tau}\widehat{x}_\tau && \text{for all } \sigma \in \mathbb{T}_{\mathcal{I}}. \end{aligned}$$

Then, there holds

$$\begin{aligned} (Ax)_i &= \sum_{\substack{(\sigma,\tau)\in\mathbb{P}_{\text{near}}\\ i\in\sigma}} (A|_{\sigma\times\tau}x|_\tau)_i + \sum_{\substack{\sigma\in\mathbb{T}_{\mathcal{I}}\\ i\in\sigma}} \Big(V_\sigma \sum_{\substack{\tau\in\mathbb{T}_{\mathcal{J}}\\ (\sigma,\tau)\in\mathbb{P}_{\text{far}}}} M_{\sigma\tau}\widehat{x}_\tau\Big)_i\\ &= \sum_{\substack{(\sigma,\tau)\in\mathbb{P}_{\text{near}}\\ i\in\sigma}} (A|_{\sigma\times\tau}x|_\tau)_i + \sum_{\substack{\sigma\in\mathbb{T}_{\mathcal{I}}\\ i\in\sigma}} (V_\sigma\widehat{y}_\sigma)_i \end{aligned}$$

That means, we split the $\mathcal{H}^2$-matrix-vector multiplication into four steps:

(i) **Forward transformation**:
Compute $\widehat{x}_\tau$ for all $\tau \in \mathbb{T}_{\mathcal{J}}$.

(ii) **Multiplication**:
Compute $\widehat{y}_\sigma$ for all $\sigma \in \mathbb{T}_{\mathcal{I}}$.

(iii) **Backward transformation**:
Compute $y_i = \sum_{\substack{\sigma\in\mathbb{T}\\ i\in\sigma}} (V_\sigma\widehat{y}_\sigma)_i$ for all $1 \le i \le m$.

(iv) **Add near field contribution**:
Update $y_i := y_i + \sum_{\substack{(\sigma,\tau)\in\mathbb{P}_{\text{near}}\\ i\in\sigma}} \big(A|_{\sigma\times\tau}x|_\tau\big)_i$ for all $1 \le i \le m$.

We stress that the vectors $\widehat{x}_\tau \in \mathbb{K}^p$ as well as $\widehat{y}_\sigma \in \mathbb{K}^p$ lead to additional storage requirements. However, this temporary memory is linear with respect to $|\mathbb{T}_{\mathcal{I}}|+|\mathbb{T}_{\mathcal{J}}| = \mathcal{O}(m+n)$. Throughout, we shall assume that these vectors are initialized by 0.

Lemma 5.4. *We assume that the vectors $\widehat{y}_\sigma \in \mathbb{K}^p$ are initialized by zero. Then, the multiplication step needs less than $2C_{\text{sparse}}p^2|\mathbb{T}_{\mathcal{I}}|$ arithmetic operations.*

Proof. For a cluster $\sigma \in \mathbb{T}_{\mathcal{I}}$, the sum over $\tau \in \mathbb{T}_{\mathcal{J}}$ with $(\sigma, \tau) \in \mathbb{P}_{\text{far}}$ has at most length C_{sparse}, i.e. we have at most C_{sparse} matrix-vector multiplications with $p \times p$ matrices, i.e. $p(2p-1)$ arithmetic operations, and C_{sparse} additions of vectors of length p. ■

Lemma 5.5. *It needs less than* $2C_{\text{sparse}}C_{\text{leaf}}^2 \min\{|\mathbb{T}_{\mathcal{I}}|, |\mathbb{T}_{\mathcal{J}}|\} \leq 2C_{\text{sparse}}C_{\text{leaf}}^2|\mathbb{T}_{\mathcal{I}}|$ *arithmetic operations to compute the near field contribution and to add it to the far-field contribution.*

Proof. We have less than $|\mathbb{P}| \leq C_{\text{sparse}} \min\{|\mathbb{T}_{\mathcal{I}}|, |\mathbb{T}_{\mathcal{J}}|\}$ near field blocks. For each block $(\sigma, \tau) \in \mathbb{P}_{\text{near}}$ we need less than $C_{\text{leaf}}(2C_{\text{leaf}} - 1)$ operations to perform the matrix-vector product $A|_{\sigma\times\tau}x|_\tau$ and we need less than C_{leaf} additions to add the result to the (already computed) far-field contribution. ■

Algorithm 5.6 (Fast Forward Transformation).

```
function FastForwardTransformation(τ, x, var x̂)
if sons(τ) = ∅
   x̂_τ := W_τ^H x|_τ
else
   for τ' ∈ sons(τ)
      FastForwardTransformation(τ',x,x̂)
      x̂_τ := x̂_τ + T_{τ'τ}^H x̂_{τ'}
   end
end
```

Lemma 5.7. *Algorithm 5.6, called by* `FastForwardTransform(`$\mathcal{J}$`,`x`,`$\widehat{x}$`)`*, computes in altogether* $6p^2|\mathbb{T}_{\mathcal{J}}| + p(2n-1)$ *arithmetic operations the vectors* $\widehat{x}_\tau$*, for all* $\tau \in \mathbb{T}$.

Proof. For each leaf $\tau \in \mathbb{T}_{\mathcal{J}}$, it needs $p(2|\tau| - 1)$ arithmetic operations to compute $\widehat{x}_\tau$. Thus, we need $p(2n-1)$ arithmetic operations to compute $\widehat{x}_\tau$ for all leaves $\tau \in \mathbb{T}$. If $\tau \in \mathbb{T}_{\mathcal{J}}$ is not a leaf, we have by definition of $\mathcal{H}^2$-matrices

$$W_\tau = \begin{pmatrix} W_{\tau'}T_{\tau'\tau} \\ W_{\tau''}T_{\tau''\tau} \end{pmatrix},$$

i.e. there holds

$$\widehat{x}_\tau = W_\tau^H x|_\tau = T_{\tau'\tau}^H W_{\tau'}^H x|_{\tau'} + T_{\tau''\tau}^H W_{\tau''}^H x|_{\tau''} = T_{\tau'\tau}^H \widehat{x}_{\tau'} + T_{\tau''\tau}^H \widehat{x}_{\tau''}.$$

This is precisely the procedure in the recursion step. Altogether, it needs $p^2 + p(2p-1) + p = 3p^2$ operations per son to update $\widehat{x}_\tau$ with $\widehat{x}_{\tau'}$. There are less than $|\mathbb{T}_{\mathcal{J}}|$ fathers in the tree. Therefore, there are less than $2|\mathbb{T}_{\mathcal{J}}|$ sons, i.e. recursion steps to be done. ■

Algorithm 5.8 (Fast Backward Transformation).

```
function FastBackwardTransformation(σ, var y, ŷ)
if sons(σ) = ∅
   y|_σ := V_σ ŷ_σ
else
   for σ' ∈ sons(σ)
      ŷ_σ' := ŷ_σ' + S_σ'σ ŷ_σ
      FastBackwardTransformation(σ',y,ŷ)
   end
end
```

Lemma 5.9. *Algorithm 5.8, called by* `FastBackwardTransformation(`$\mathcal{I}$`,`y`,`$\widehat{y}$`)`*, needs less than* $4p^2|\mathbb{T}_\mathcal{I}| + m(2p-1)$ *arithmetic operations to finish the computation of the far field contribution.*

Proof. To understand why Algorithm 5.8 works properly, let σ, σ' be clusters with $i \in \sigma' \in \operatorname{sons}(\sigma)$. Using that $V_\sigma|_{\sigma'} = V_{\sigma'} S_{\sigma'\sigma}$, we obtain

$$\begin{aligned} y_i := \sum_{\substack{\sigma \in \mathbb{T}_\mathcal{I} \\ i \in \sigma}} (V_\sigma \widehat{y}_\sigma)_i &= (V_\sigma \widehat{y}_\sigma)_i + (V_{\sigma'} \widehat{y}_{\sigma'})_i + \sum_{\substack{\tilde{\sigma} \in \mathbb{T} \setminus \{\sigma, \sigma'\} \\ i \in \tilde{\sigma}}} (V_{\tilde{\sigma}} \widehat{y}_{\tilde{\sigma}})_i \\ &= \big(V_{\sigma'}(S_{\sigma'\sigma} \widehat{y}_\sigma + \widehat{y}_{\sigma'})\big)_i + \sum_{\substack{\tilde{\sigma} \in \mathbb{T} \setminus \{\sigma, \sigma'\} \\ i \in \tilde{\sigma}}} (V_{\tilde{\sigma}} \widehat{y}_{\tilde{\sigma}})_i. \end{aligned}$$

This explains the recursive structure in the algorithm. Analogously to the fast forward transformation, we have $m(2p-1)$ arithmetic operations for the computation of matrix-vector multiplications on the leaf level of $\mathbb{T}_\mathcal{I}$. If $\sigma \in \mathbb{T}$ is not a leaf, the recursion step needs $p(2p-1)+p = 2p^2$ operations per son, i.e. altogether less than $4p^2|\mathbb{T}_\mathcal{I}|$ arithmetic operations. ■

Finally, we have proved the following proposition which shows that the complexity for the matrix-vector multiplication is independent on how good the cluster tree is: The depth of $\mathbb{T}$ does not enter the algorithmic complexity.

Proposition 5.10. *By use of the introduced algorithms, the number* N_{MVM} *of arithmetic operations for the matrix-vector multiplication with an* $\mathcal{H}^2$*-matrix* $A \in \mathbb{K}^{m \times n}$ *is bounded by*

$$N_{\mathrm{MVM}} \leq \big(2C_{\mathrm{sparse}}(p^2 + C_{\mathrm{leaf}}^2) + 10p^2\big) \max\{|\mathbb{T}_\mathcal{I}|, |\mathbb{T}_\mathcal{J}|\} + 2p(m+n) = \mathcal{O}(p^2 N)$$

and thus linear with respect to the problem size $N = m + n$. ■