Max-Planck-Institut
für Mathematik
in den Naturwissenschaften
Leipzig

$\mathcal{H}^2$-matrices — Multilevel methods for the approximation of integral operators

by

Steffen Börm

Preprint no.: 7 2003

$\mathcal{H}^2$-matrices — Multilevel methods for the approximation of integral operators

Steffen Börm*

January 31, 2003

Multigrid methods are typically used to solve partial differential equations, i.e., they approximate the inverse of the corresponding partial differential operators. At least for elliptic PDEs, this inverse can be expressed in the form of an integral operator by Green's theorem.

This implies that multigrid methods approximate certain integral operators, so it is straightforward to look for variants of multigrid methods that can be used to approximate more general integral operators.

$\mathcal{H}^2$-matrices combine a multigrid-like structure with ideas from panel clustering algorithms in order to provide a very efficient method for discretizing and evaluating the integral operators found, e.g., in boundary element applications.

AMS Subject Classification: 65F05, 65F30, 65F50, 65N50
Key words: Hierarchical matrices, data-sparse approximations

1 Introduction

We consider the integral operator $\mathcal{K}$ given by

$$\mathcal{K}[u](x) := \int_\Gamma \kappa(x,y)u(y)\,dy,$$

where $\Gamma \subseteq \mathbb{R}^d$ is a domain or a submanifold and $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a kernel function.

If we discretize $\mathcal{K}$ by a Galerkin method using a basis $(\Psi_i)_{i\in\mathcal{I}}$ of functions, we have to construct the matrix $\mathbf{K} \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ given by

$$\mathbf{K}_{ij} := \int_\Gamma \Psi_i(x) \int_\Gamma \kappa(x,y)\Psi_j(y)\,dy\,dx \qquad \text{for all } i,j \in \mathcal{I}.$$

*Max-Planck-Institute for Mathematics in the Sciences, Inselstraße 22–26, 04103 Leipzig, Germany

In typical applications, the kernel function $\kappa(\cdot,\cdot)$ has non-local support, so the matrix $\mathbf{K}$ will be densely populated even if we use a finite element basis, and this implies that the complexity of algorithms treating this matrix will be very high.

There are different approaches to this problem: For very simple geometries and simple kernel functions, the fast Fourier transform can be used to reduce $\mathbf{K}$ to diagonal form. For more complicated geometries, we can replace $\mathbf{K}$ by an approximation using, e.g., the panel clustering technique [14], multipole expansions [15] or wavelet compression schemes [6].

Our approach is based on multigrid techniques: The matrices corresponding to multilevel preconditioners for elliptic partial differential equations are densely populated, too, but can nevertheless be implemented efficiently. Therefore our goal is to use a structure similar to that of multigrid algorithms in order to evaluate integral operators: The structure of $\mathcal{H}^2$-matrices (cf. [13]), a specialization of hierarchical matrices (cf. [10, 12, 11, 1]).

In the next section, we will analyze a simple multigrid method in order to find those basic structures that can be used for our purpose. We will then generalize these structures and use the result to derive our algorithm for evaluating discretized integral operators. The presentation of an alternative method for creating $\mathcal{H}^2$-matrices and numerical results conclude the paper.

2 Basic structure of multigrid methods

Let us consider the simple example of the multigrid method for the linear equation

$$\mathbf{A}x = b.$$

We fix a hierarchy $(\mathbf{A}^\ell)_{\ell=0}^L$ of stiffness matrices, a hierarchy $(\mathbf{S}^\ell)_{\ell=0}^L$ of smoothing matrices and a hierarchy $(\mathbf{P}^\ell)_{\ell=0}^{L-1}$ of prolongation matrices. We denote the set of degrees of freedom on a level $\ell \in \{0,\dots,L\}$ by $\mathcal{I}^\ell$, i.e., we have $\mathbf{A}^\ell, \mathbf{S}^\ell \in \mathbb{R}^{\mathcal{I}^\ell \times \mathcal{I}^\ell}$ for all $\ell \in \{0,\dots,L\}$ and $\mathbf{P}^\ell \in \mathbb{R}^{\mathcal{I}^{\ell+1}\times\mathcal{I}^\ell}$ for all $\ell \in \{0,\dots L-1\}$.

We simplify the usual multigrid algorithm by leaving out the post-smoothing step and using only one pre-smoothing step:

```
procedure MG(ℓ, x_ℓ, b_ℓ);
begin
  d^ℓ := b^ℓ − A^ℓ x^ℓ; x^ℓ := x^ℓ + S^ℓ d^ℓ;
  if ℓ > 0 then begin
    d^ℓ := b^ℓ − A^ℓ x^ℓ;                                  (*)
    x^{ℓ−1} := 0; MG(ℓ − 1, x^{ℓ−1}, P^{ℓ−1}^⊤ d^ℓ);
    x^ℓ := x^ℓ + P^{ℓ−1} x^{ℓ−1}
  end
end
```

If the second assignment to d^ℓ (marked by (*) in the above algorithm) is present, we have the common multiplicative multigrid algorithm, if this assignment is left out, we

get an additive method. The advantage of the additive approach is the simplicity of the resulting iteration: Let $\tilde{x}^L$ denote an initial guess. The defect d^L on the finest level is given by $b - \mathbf{A}^\ell \tilde{x}^L$. In the additive setting, we find

$$d^\ell = {\mathbf{P}^\ell}^\top \cdots {\mathbf{P}^{L-1}}^\top d^L = {\widehat{\mathbf{P}}^\ell}^\top d^L \quad \text{for all } \ell \in \{0, \dots, L-1\},$$

where we set $\widehat{\mathbf{P}}^\ell := \mathbf{P}^{L-1} \cdots \mathbf{P}^\ell$ and $\widehat{\mathbf{P}}^L := \mathbf{I}$. This implies

$$x^L = \sum_{\ell=0}^{L} \widehat{\mathbf{P}}^\ell \mathbf{S}^\ell d^\ell = \left(\sum_{\ell=0}^{L} \widehat{\mathbf{P}}^\ell \mathbf{S}^\ell {\widehat{\mathbf{P}}^\ell}^\top \right) d_L,$$

so the additive multigrid iteration can be written in the form

$$x^L := \mathbf{N}(b - \mathbf{A}^L \tilde{x}^L) \quad \text{with} \quad \mathbf{N} := \sum_{\ell=0}^{L} \widehat{\mathbf{P}}^\ell \mathbf{S}^\ell {\widehat{\mathbf{P}}_\ell}^\top.$$

We assume that the index sets are disjoint, i.e., that $\ell \neq \ell' \Rightarrow \mathcal{I}^\ell \cap \mathcal{I}^{\ell'} = \emptyset$ holds for $\ell, \ell' \in \{0, \dots, L\}$. For each $i \in \mathcal{I}^\ell$, we define $e^i \in \mathbb{R}^{\mathcal{I}^\ell}$ by $e^i_j := \delta_{ij}$ and set $v^i := \widehat{\mathbf{P}}^\ell e^i$. Let

$$P^\ell := \{(i,j) \in \mathcal{I}^\ell \times \mathcal{I}^\ell \ : \ (\mathbf{S}^\ell)_{ij} \neq 0\} \quad \text{for } \ell \in \{0, \dots, L\}$$

and define $P := \bigcup_{\ell=0}^{L} P^\ell$. The matrix $\mathbf{N}$ can be written in the form

$$\mathbf{N} = \sum_{(i,j) \in P} v^i \mathbf{S}^\ell_{ij} {v^j}^\top, \tag{1}$$

i.e., as a sum of rank 1 matrices.

Let $i \in \mathcal{I}^\ell$ for $\ell \in \{0, \dots, L-1\}$. Due to the definition of $\widehat{\mathbf{P}}^\ell$, we have

$$v^i = \widehat{\mathbf{P}}^\ell e^i = \widehat{\mathbf{P}}^{\ell+1} \mathbf{P}^\ell e^i = \widehat{\mathbf{P}}^{\ell+1} \sum_{j \in \mathcal{I}^{\ell+1}} \mathbf{P}^\ell_{ji} e^j = \sum_{j \in \mathcal{I}^{\ell+1}} \mathbf{P}^\ell_{ji} v^j = \sum_{j \in \mathfrak{S}^i} \mathbf{P}^\ell_{ji} v^j, \tag{2}$$

where $\mathfrak{S}^i := \{j \in \mathcal{I}^{\ell+1} \ : \ \mathbf{P}^\ell_{ji} \neq 0\}$. This means that the vectors v^i corresponding to lower levels of the hierarchy can be expressed in terms of vectors v^j corresponding to higher levels of the hierarchy. In typical multigrid schemes, the prolongation matrices $\mathbf{P}^\ell$ are sparse, i.e., the set $\mathfrak{S}^i$ contains only a small number of elements.

We assume that each degree of freedom $i \in \mathcal{I}^\ell$, $\ell \in \{0, \dots, L\}$, corresponds to a subdomain $\Omega_i \subseteq \mathbb{R}^d$ of d-dimensional space, e.g., the support of the associated basis function in finite element methods. In typical multigrid algorithms, the smoothing matrices $\mathbf{S}^\ell$ are local operators, i.e., we have

$$\mathbf{S}^\ell_{ij} \neq 0 \Rightarrow \Omega_i \cap \Omega_j \neq \emptyset \quad \text{for all } i, j \in \mathcal{I}^\ell, \ell \in \{0, \dots, L\}.$$

This implies

$$\mathbf{S}^\ell_{ij} \neq 0 \Rightarrow \operatorname{dist}(\Omega_i, \Omega_j) \leq C \max\{\operatorname{diam}(\Omega_i), \operatorname{diam}(\Omega_j)\} \tag{3}$$

for *any* constant $C \in \mathbb{R}_{>0}$ and $i, j \in \mathcal{I}^\ell$, $\ell \in \{0, \dots, L\}$.

The sum (1) can be transformed in such a way that the upper bound in (3) is complemented by a lower bound.

For a given constant $C' \in \mathbb{R}_{>0}$ and a given level $\ell \in \{0, \dots, L-1\}$, we define the indicator matrix $\delta^\ell \in \mathbb{R}^{\mathcal{I}^\ell \times \mathcal{I}^\ell}$ by marking all pairs of indices satisfying the reverse of (3):

$$\delta_{ij}^\ell = \begin{cases} 1 & \text{if } \max\{\operatorname{diam}(\Omega_i), \operatorname{diam}(\Omega_j)\} \leq C' \operatorname{dist}(\Omega_i, \Omega_j) \\ 0 & \text{otherwise.} \end{cases}$$

The operator

$$\chi^\ell : \mathbb{R}^{\mathcal{I}^\ell \times \mathcal{I}^\ell} \to \mathbb{R}^{\mathcal{I}^\ell \times \mathcal{I}^\ell}, \quad \mathbf{X} \mapsto (\delta_{ij}^\ell \mathbf{X}_{ij})_{i,j \in \mathcal{I}^\ell},$$

removes all unwanted coefficients. Now we can recursively define the desired transformation: For $\ell \in \{0, \dots, L-1\}$, we can split a given matrix $\mathbf{S}^\ell$ into the part $\chi^\ell \mathbf{S}^\ell$ having non-zero coefficients only for indices satisfying our inequality and the remainder $\mathbf{S}^\ell - \chi^\ell \mathbf{S}^\ell$ that can be transformed to the next finer level due to (2):

$$\begin{aligned} \widehat{\mathbf{P}}^\ell \mathbf{S}^\ell \widehat{\mathbf{P}}^{\ell\top} &= \widehat{\mathbf{P}}^\ell \chi^\ell \mathbf{S}^\ell \widehat{\mathbf{P}}^{\ell\top} + \widehat{\mathbf{P}}^\ell (\mathbf{S}^\ell - \chi^\ell \mathbf{S}^\ell) \widehat{\mathbf{P}}^{\ell\top} \\ &= \widehat{\mathbf{P}}^\ell \chi^\ell \mathbf{S}^\ell \widehat{\mathbf{P}}^{\ell\top} + \widehat{\mathbf{P}}^{\ell+1} \left[\mathbf{P}^\ell (\mathbf{S}^\ell - \chi^\ell \mathbf{S}^\ell) \mathbf{P}^{\ell\top} \right] \widehat{\mathbf{P}}^{\ell+1\top}. \end{aligned}$$

We apply this procedure recursively by defining

$$\widehat{\mathbf{S}}^0 := \mathbf{S}^0, \quad \widehat{\mathbf{S}}^{\ell+1} := \mathbf{S}^{\ell+1} + \mathbf{P}^\ell (\widehat{\mathbf{S}}^\ell - \chi^\ell \widehat{\mathbf{S}}^\ell) \mathbf{P}^{\ell\top} \quad \text{for all } \ell \in \{0, \dots, L-1\}.$$

The matrix $\widehat{\mathbf{S}}^{\ell+1}$ consists of the matrix $\mathbf{S}^{\ell+1}$ and the prolongation of that part of $\widehat{\mathbf{S}}^\ell$ that did not match our condition. We define

$$\tilde{\mathbf{S}}^\ell := \begin{cases} \chi^\ell \widehat{\mathbf{S}}^\ell & \text{if } \ell < L \\ \widehat{\mathbf{S}}^\ell & \text{otherwise,} \end{cases}$$

and this implies that

$$\widehat{\mathbf{P}}^{\ell+1} \widehat{\mathbf{S}}^{\ell+1} \widehat{\mathbf{P}}^{\ell+1\top} - \widehat{\mathbf{P}}^\ell \widehat{\mathbf{S}}^\ell \widehat{\mathbf{P}}^{\ell\top} + \widehat{\mathbf{P}}^\ell \tilde{\mathbf{S}}^\ell \widehat{\mathbf{P}}^{\ell\top} = \widehat{\mathbf{P}}^{\ell+1} \mathbf{S}^{\ell+1} \widehat{\mathbf{P}}^{\ell+1\top}$$

holds for all $\ell \in \{0, \dots, L-1\}$. This means

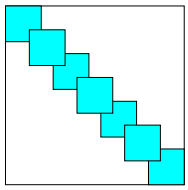
$$\begin{aligned} \mathbf{N} &= \sum_{\ell=0}^{L} \widehat{\mathbf{P}}^\ell \mathbf{S}^\ell \widehat{\mathbf{P}}^{\ell\top} = \widehat{\mathbf{P}}^0 \mathbf{S}^0 \widehat{\mathbf{P}}^{0\top} + \sum_{\ell=0}^{L-1} \widehat{\mathbf{P}}^{\ell+1} \mathbf{S}^{\ell+1} \widehat{\mathbf{P}}^{\ell+1\top} \\ &= \widehat{\mathbf{P}}^0 \mathbf{S}^0 \widehat{\mathbf{P}}^{0\top} + \sum_{\ell=0}^{L-1} (\widehat{\mathbf{P}}^{\ell+1} \widehat{\mathbf{S}}^{\ell+1} \widehat{\mathbf{P}}^{\ell+1\top} - \widehat{\mathbf{P}}^\ell \widehat{\mathbf{S}}^\ell \widehat{\mathbf{P}}^{\ell\top} + \widehat{\mathbf{P}}^\ell \tilde{\mathbf{S}}^\ell \widehat{\mathbf{P}}^{\ell\top}) \\ &= \widehat{\mathbf{P}}^0 \mathbf{S}^0 \widehat{\mathbf{P}}^{0\top} + \widehat{\mathbf{P}}^L \widehat{\mathbf{S}}^L \widehat{\mathbf{P}}^{L\top} - \widehat{\mathbf{P}}^0 \widehat{\mathbf{S}}^0 \widehat{\mathbf{P}}^{0\top} + \sum_{\ell=0}^{L-1} \widehat{\mathbf{P}}^\ell \tilde{\mathbf{S}}^\ell \widehat{\mathbf{P}}^{\ell\top} = \sum_{\ell=0}^{L} \widehat{\mathbf{P}}^\ell \tilde{\mathbf{S}}^\ell \widehat{\mathbf{P}}^{\ell\top}, \end{aligned}$$

so we have found an alternative representation of $\mathbf{N}$ that still has the form (1) such that

$$\tilde{\mathbf{S}}_{ij}^{\ell} \neq 0 \Rightarrow \max\{\operatorname{diam}(\Omega_i), \operatorname{diam}(\Omega_j)\} \leq C' \operatorname{dist}(\Omega_i, \Omega_j)$$

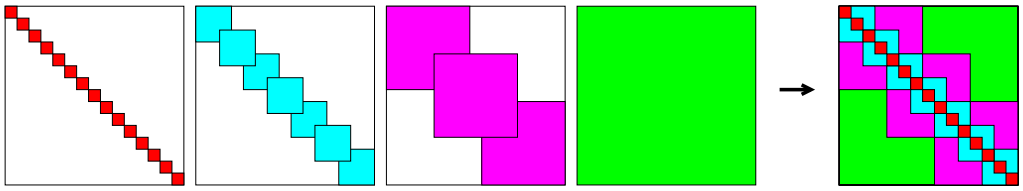
holds for all $\ell \in \{0, \dots, L-1\}$. In standard situations, the $\tilde{\mathbf{S}}^{\ell}$ of the new representation will be sparse and the diameters and the distance will be proportional.

Example 2.1 (One-dimensional case) *We consider a one-dimensional partial differential equation discretized by the common piecewise linear nodal basis functions on a dyadic grid hierarchy. We use a Jacobi smoother, so the smoothing matrices $\mathbf{S}^{\ell}$ are diagonal. Due to the choice of basis functions, the matrices $\widehat{\mathbf{P}}^{\ell}\mathbf{S}^{\ell}{\widehat{\mathbf{P}}^{\ell}}^{\top}$ take the following form:*



Each diagonal entry of the matrix $\mathbf{S}^{\ell}$ represents a nodal basis function, and the matrix $\widehat{\mathbf{P}}^{\ell}$ corresponds to the interpolation of this basis function in the grid points on the finest level of the hierarchy. Since $\mathbf{S}^{\ell}$ is the composition of its diagonal entries, the matrix $\widehat{\mathbf{P}}^{\ell}\mathbf{S}^{\ell}{\widehat{\mathbf{P}}^{\ell}}^{\top}$ is a composition of matrix blocks of rank 1.

For the matrix $\mathbf{N}$, we combine the interpolated smoothing matrices on all levels and get the following representation:



We can see that the size of the non-diagonal blocks is proportional to their distance from the diagonal.

3 Generalization

Let $\mathcal{I}^* := \bigcup_{\ell=0}^{L} \mathcal{I}^{\ell}$. In the previous section, we have seen that the matrix $\mathbf{N}$ corresponding to an additive multigrid preconditioner has the following properties: For each $i \in \mathcal{I}^*$, there is a vector $v^i \in \mathbb{R}^{\mathcal{I}^L}$ with an associated subdomain $\Omega_i \subseteq \mathbb{R}^d$ such that the matrix $\mathbf{N}$ is a sum of rank-1-blocks of the form $v^i s^{ij} {v^j}^{\top}$ with

$$\operatorname{dist}(\Omega_i, \Omega_j) \sim \max\{\operatorname{diam}(\Omega_i), \operatorname{diam}(\Omega_j)\}.$$

If $\ell \in \{0, \ldots, L-1\}$, each $i \in \mathcal{I}^\ell$ can be expressed in the form

$$v^i = \sum_{j \in \mathfrak{S}^i} v^j p_{ji}$$

for coefficients $p_{ji} \in \mathbb{R}$ and a subset $\mathfrak{S}^i \subseteq \mathcal{I}^{\ell+1}$.

In order to derive a similar algorithm for the evaluation of integral operators, we have to generalize this structure.

3.1 Cluster tree

In the multigrid setting, we have introduced subdomains Ω^i corresponding to degrees i of freedom. In typical boundary element applications, a hierarchy of degrees of freedom is not readily available, but it is still possible to construct a hierarchy of subdomains:

Definition 3.1 (Cluster tree) *Let $\mathcal{T}_I$ be a tree satisfying*

1. *Γ is the root of $\mathcal{T}_I$.*
2. *If a node τ of $\mathcal{T}_I$ is not a leaf, it is the union of its sons $\mathfrak{S}^\tau$ and $\#\mathfrak{S}^\tau \leq C_{\mathrm{sons}}$ holds for a constant $C_{\mathrm{sons}} \in \mathbb{N}$.*
3. *If a node τ of $\mathcal{T}_I$ is a leaf, the set*

$$\hat{\tau} := \{i \in \mathcal{I} \ : \ \operatorname{supp} \Psi_i \cap \tau \neq \emptyset\}$$

satisfies $\#\hat{\tau} \leq C_{\mathrm{leaf}}$ for a constant $C_{\mathrm{leaf}} \in \mathbb{N}$.

Then $\mathcal{T}_I$ is called a cluster tree*, the set of its nodes is denoted by T_I and its elements $\tau \in T_I$ are called* clusters*.*

We note that this definition implies $\tau \subseteq \Gamma$ for all nodes of $\mathcal{T}_I$. Simple algorithms for constructing suitable cluster trees for relatively general finite element grids can be found in [8, 9, 1].

3.2 Block partition

We have seen that the multigrid operator can be expressed in the form of a sum of terms of the form $v^i s^{ij} {v^j}^\top$ that satisfy $\max\{\operatorname{diam}(\Omega_i), \operatorname{diam}(\Omega_j)\} \sim C' \operatorname{dist}(\Omega_i, \Omega_j)$.

We make use of this condition to create a similar structure: A pair (τ, σ) of clusters is called *η-admissible* for a parameter $\eta \in \mathbb{R}_{>0}$ if

$$\max\{\operatorname{diam}(\tau), \operatorname{diam}(\sigma)\} \leq \eta \operatorname{dist}(\tau, \sigma) \tag{4}$$

holds. Using this criterion, the following algorithm constructs the necessary blocks:

```
procedure BuildPartition(τ, σ, var P);
begin
  if (τ,σ) is η-admissible then P := P ∪ {(τ,σ)}                { admissible block }
  else if 𝔖^τ = ∅ then
    if 𝔖^σ = ∅ then
      P := P ∪ {(τ,σ)}                                          { no splitting possible }
    else
      for σ' ∈ 𝔖^σ do BuildPartition(τ, σ', P)                  { split σ only }
  else
    if 𝔖^σ = ∅ then
      for τ' ∈ 𝔖^τ do BuildPartition(τ', σ, P)                  { split τ only }
    else
      for τ' ∈ 𝔖^τ, σ' ∈ 𝔖^σ do BuildPartition(τ', σ', P)        { split τ and σ }
end
```

We collect the admissible blocks of P in P_{far} (they are referred to as *far-field blocks*, since they are far from the diagonal) and the remaining blocks in P_{near} (these are called *near-field blocks*):

$$P_{\mathrm{far}} := \{(\tau,\sigma) \in P \ : \ (\tau,\sigma) \text{ is } \eta\text{-admissible}\}, \quad P_{\mathrm{near}} := P \setminus P_{\mathrm{far}}.$$

3.3 Increased rank

The blocks of the multigrid operator $\mathbf{N}$ are stored in the form $v^i s^{ij} v^{j\top}$ and have rank 1. Since the matrix $\mathbf{N}$ is only a rough approximation of the inverse of the stiffness matrix, we have to use multiple iteration steps in order to reach a sufficient accuracy of the solution.

We do not have the possibility of using an iterative process in order to evaluate a general integral operator, therefore we have to look for an alternative way of improving the accuracy. The solution is to increase the rank: For each cluster $\tau \in T_I$, we fix a rank k^τ and introduce a matrix $\mathbf{V}^\tau \in \mathbb{R}^{\mathcal{I}\times k^\tau}$ describing a *cluster basis* as a generalization of the vectors v^i. The coefficients s^{ij} are replaced by a matrix $\mathbf{S}^{\tau,\sigma} \in \mathbb{R}^{k^\tau \times k^\sigma}$ for blocks $(\tau,\sigma) \in P$.

Now we can introduce the generalization of the structure from (1) that we will use for the approximation of the matrix $\mathbf{K}$:

$$\tilde{\mathbf{K}} := \sum_{(\tau,\sigma)\in P_{\mathrm{far}}} \mathbf{V}^\tau \mathbf{S}^{\tau,\sigma} \mathbf{V}^{\sigma\top} + \sum_{(\tau,\sigma)\in P_{\mathrm{near}}} \hat{\mathbf{S}}^{\tau,\sigma}. \tag{5}$$

A matrix given in this form is called a *uniform hierarchical matrix*.

There is still one thing missing: For the vectors v^i, the equation (2) holds, i.e., they are nested. This property is crucial for the efficiency of multigrid methods, so we have to make sure that the generalized method preserves it. We do this by requiring that there are matrices $\mathbf{B}^{\tau',\tau} \in \mathbb{R}^{k^{\tau'}\times k^\tau}$ for all $\tau \in T_I$ and $\tau' \in \mathfrak{S}^\tau$ such that

$$\mathbf{V}^\tau = \sum_{\tau'\in\mathfrak{S}^\tau} \mathbf{V}^{\tau'} \mathbf{B}^{\tau',\tau} \tag{6}$$

holds. A uniform hierarchical matrix whose cluster bases satisfy the condition (6) is called an $\mathcal{H}^2$-*matrix*[1] [13].

We have to rearrange the multigrid algorithm in order to adapt it to the generalized structure: The restriction of the input vector to all grid levels has to be done separately since we traverse the cluster tree from the coarse to the finer levels, but information has to be passed in the reverse direction:

```
procedure Restriction(τ);
begin
  if 𝔖^τ = ∅ then                                  { highest level }
    x^τ := V^τ^⊤ x                                 { apply V^τ directly }
  else begin                                       { intermediate level }
    x^τ := 0
    for τ' ∈ 𝔖^τ do begin
      Restriction(τ'); x^τ := B^τ',τ^⊤ x^τ'        { use (6) }
    end
  end
end
```

This algorithm computes the coefficients $x^\tau := \mathbf{V}^{\tau\top} x$ for all clusters $\tau \in T_I$. The following procedure performs the rest of the matrix-vector multiplication:

```
procedure Multiplication(τ);
begin
  y^τ := 0;
  for σ ∈ T_I with (τ,σ) ∈ P_far do y^τ := y^τ + S^τ,σ x^σ;   { multiply by S^τ,σ }
  if 𝔖^τ = ∅ then                                  { highest level }
    y := y + V^τ y^τ
  else begin                                       { intermediate level }
    for τ' ∈ 𝔖^τ do begin
      y^τ' := y^τ' + B^τ',τ y^τ; Multiplication(τ')   { use (6) }
    end
  end
end
```

Combining both procedures gives us the fast matrix-vector multiplication algorithm:

```
procedure MVM(x, var y);
begin
  Restriction(Γ);                                  { compute x^τ }
  y := 0;
  Multiplication(Γ);                               { compute far-field part }
  for (τ,σ) ∈ P_near do y := y + Ŝ^τ,σ x           { add near-field part }
end
```

[1]The name $\mathcal{H}^2$-matrix is due to the fact that we are using *two* hierarchies: The cluster tree and the hierarchy of the cluster bases.

For typical cluster trees and block partitions, the matrix-vector multiplication requires $\mathcal{O}(nk_{\max})$ operations, where $n := \#\mathcal{I}$ is the number of degrees of freedom and where $k_{\max} := \max\{k^\tau \ : \ \tau \in T_I\}$ is the maximal rank. This means that our generalized algorithm, just as a usual multigrid algorithm, has optimal complexity in n.

4 $\mathcal{H}^2$-matrix approximation of integral operators

We will now apply the $\mathcal{H}^2$-matrix technique to the problem of approximating integral operators: We have to construct appropriate matrices $\mathbf{V}^\tau$, $\mathbf{B}^{\tau',\tau}$ and $\mathbf{S}^{\tau,\sigma}$ such that the resulting $\mathcal{H}^2$-matrix approximates the discretized integral operators $\mathbf{K}$.

4.1 Kernel approximation

We recall that the matrix $\mathbf{K}$ is defined by

$$\mathbf{K}_{ij} = \int_\Gamma \Psi_i(x) \int_\Gamma \kappa(x,y)\Psi_j(y)\,dy\,dx.$$

If the kernel function $\kappa(\cdot,\cdot)$ is *degenerate*, i.e., if it has the form

$$\kappa(x,y) = \sum_{\nu=1}^{k}\sum_{\mu=1}^{l} s_{\nu,\mu} f_\nu(x) g_\mu(y)$$

for functions $(f_\nu)_{\nu=1}^k$ and $(g_\mu)_{\mu=1}^l$ and coefficients $(s_{\nu,\mu})_{\nu,\mu=1}^{k,l}$, we find

$$\mathbf{K}_{ij} = \sum_{\nu=1}^{k}\sum_{\mu=1}^{l} s_{\nu,\mu} \underbrace{\int_\Gamma \Psi_i(x) f_\nu(x)\,dx}_{=:F_{i\nu}} \underbrace{\int_\Gamma \Psi_j(y) g_\mu(y)\,dy}_{=:G_{j\mu}} = (FSG^\top)_{ij},$$

i.e., the matrix $\mathbf{K}$ has the desired form (5).

Unfortunately, typical kernel functions are not degenerate, so we will not be able to use the above representation globally. However, we can use it *locally*: The standard kernel functions are *asymptotically smooth*, i.e., they satisfy an estimate of the form

$$|\partial_x^\alpha \partial_y^\beta \kappa(x,y)| \leq C(\alpha+\beta)!(c_0\|x-y\|)^{-g-|\alpha|-|\beta|} \tag{7}$$

for constants $C, c_0 \in \mathbb{R}_{>0}$, a parameter $g \in \mathbb{R}_{>0}$ denoting the degree of singularity at the diagonal $x = y$ and arbitrary multi-indices $\alpha, \beta \in \mathbb{N}_0^d$. This estimate implies that the kernel is smooth in domains that are removed from the diagonal.

A smooth function can be approximated by polynomial interpolation, and any polynomial is automatically degenerate, so we can construct degenerate approximations of the kernel function by interpolation (cf. [7, 3]).

In order to get a simple algorithm, we will use tensor-product interpolation, and for this we need axis-parallel domains. Therefore we fix an axis-parallel box $Q^\tau \subseteq \mathbb{R}^d$ for

each cluster $\tau \in T_I$ satisfying $\tau \subseteq Q^\tau$. Due to this latter property, the boxes Q^τ are called *bounding boxes*.

In order to find an approximation of the kernel function for a block $(\tau, \sigma) \in P$, we will use interpolation on the domain $Q^\tau \times Q^\sigma$, so we have to ensure that the kernel function is smooth on this domain. We do this by replacing the admissibility condition (4) by the stronger condition

$$\max\{\operatorname{diam}(Q^\tau), \operatorname{diam}(Q^\sigma)\} \leq \eta \operatorname{dist}(Q^\tau, Q^\sigma). \tag{8}$$

We fix an interpolation operator $\mathfrak{I}^\tau$ of order m^τ for each bounding box Q^τ with corresponding interpolation points $(x_\nu^\tau)_{\nu \in M^\tau}$ and Lagrange polynomials $(\mathcal{L}_\nu^\tau)_{\nu \in M^\tau}$.

For each admissible block $(\tau, \sigma) \in P_{\text{far}}$, we define the kernel approximation

$$\tilde{\kappa}^{\tau,\sigma}(x,y) := (\mathfrak{I}^\tau \otimes \mathfrak{I}^\sigma)[\kappa](x,y) = \sum_{\nu \in M^\tau} \sum_{\mu \in M^\sigma} \kappa(x_\nu^\tau, x_\mu^\sigma) \mathcal{L}_\nu^\tau(x) \mathcal{L}_\mu^\sigma(y).$$

The partition P describes a decomposition of the set $\Gamma \times \Gamma$ into admissible and non-admissible subdomains. We define the approximation of the kernel function $\kappa(\cdot,\cdot)$ by replacing the original kernel function by its local approximations in all far-field blocks:

$$\tilde{\kappa}(x,y) := \begin{cases} \tilde{\kappa}^{\tau,\sigma}(x,y) & \text{if } x \in \tau, y \in \sigma, (\tau,\sigma) \in P_{\text{far}} \\ \kappa(x,y) & \text{otherwise.} \end{cases}$$

4.2 Matrix approximation

The approximate matrix $\tilde{\mathbf{K}}$ is derived by discretizing $\tilde{\kappa}(\cdot,\cdot)$:

$$\begin{aligned} \tilde{\mathbf{K}}_{ij} &= \int_\Gamma \Psi_i(x) \int_\Gamma \tilde{\kappa}(x,y) \Psi_j(y)\, dy\, dx = \sum_{(\tau,\sigma) \in P} \int_\tau \Psi_i(x) \int_\sigma \tilde{\kappa}(x,y) \Psi_j(y)\, dy\, dx \\ &= \sum_{(\tau,\sigma) \in P_{\text{far}}} \int_\tau \Psi_i(x) \int_\sigma \tilde{\kappa}^{\tau,\sigma}(x,y) \Psi_j(y)\, dy\, dx \\ &\quad + \sum_{(\tau,\sigma) \in P_{\text{near}}} \int_\tau \Psi_i(x) \int_\sigma \kappa(x,y) \Psi_j(y)\, dy\, dx \\ &= \sum_{(\tau,\sigma) \in P_{\text{far}}} \mathbf{V}^\tau \mathbf{S}^{\tau,\sigma} \mathbf{V}^{\sigma\top} + \sum_{(\tau,\sigma) \in P_{\text{near}}} \hat{\mathbf{S}}^{\tau,\sigma}, \end{aligned}$$

where the matrices $\mathbf{V} \in \mathbb{R}^{\mathcal{I} \times M^\tau}$, $\mathbf{S}^{\tau,\sigma} \in \mathbb{R}^{M^\tau \times M^\sigma}$ and $\hat{\mathbf{S}}^{\tau,\sigma} \in \mathbb{R}^{\mathcal{I} \times \mathcal{I}}$ are defined by

$$\mathbf{V}_{i\nu}^\tau := \int_\tau \mathcal{L}_\nu^\tau(x) \Psi_i(x)\, dx, \quad \mathbf{S}_{\nu\mu}^{\tau,\sigma} := \kappa(x_\nu^\tau, x_\mu^\sigma) \quad \text{and}$$

$$\hat{\mathbf{S}}_{ij}^{\tau,\sigma} := \int_\tau \Psi_i(x) \int_\sigma \kappa(x,y) \Psi_j(y)\, dy\, dx.$$

We note that $\tau \cap \operatorname{supp} \Psi_i = \emptyset$ implies $\mathbf{V}_{i\nu}^\tau = 0$ for all $i \in \mathcal{I}$ and $\nu \in M^\tau$, so a large number of rows of $\mathbf{V}^\tau$ will be zero. By the same reasoning, we see that most of the entries of the near-field matrices $\hat{\mathbf{S}}^{\tau,\sigma}$ will be zero, too.

Let $\tau \in T_I$ and $\tau' \in \mathfrak{S}^\tau$. If $m^\tau \leq m^{\tau'}$, we have $\mathcal{L}_\nu^\tau = \mathfrak{I}^{\tau'}[\mathcal{L}_\nu^\tau]$, and therefore

$$\begin{aligned}
\mathbf{V}_{i\nu}^\tau &= \int_\tau \mathcal{L}_\nu^\tau(x)\Psi_i(x)\,dx = \sum_{\tau'\in\mathfrak{S}^\tau} \int_{\tau'} \mathcal{L}_\nu^\tau(x)\Psi_i(x)\,dx \\
&= \sum_{\tau'\in\mathfrak{S}^\tau} \int_{\tau'} \mathfrak{I}^{\tau'}[\mathcal{L}_\nu^\tau](x)\Psi_i(x)\,dx = \sum_{\tau'\in\mathfrak{S}^\tau} \sum_{\nu'\in M^{\tau'}} \mathcal{L}_\nu^\tau(x_{\nu'}^{\tau'}) \int_{\tau'} \mathcal{L}_{\nu'}^{\tau'}(x)\Psi_i(x)\,dx \\
&= \left(\sum_{\tau'\in\mathfrak{S}^\tau} \mathbf{V}^{\tau'}\mathbf{B}^{\tau',\tau} \right)_{i\nu}
\end{aligned}$$

holds for $\mathbf{B}^{\tau',\tau} \in \mathbb{R}^{M^{\tau'}\times M^\tau}$ defined by

$$\mathbf{B}_{\nu'\nu}^{\tau',\tau} := \mathcal{L}_\nu^\tau(x_{\nu'}^{\tau'}),$$

so our cluster bases satisfy condition (6), and this implies that $\tilde{\mathbf{K}}$ is an $\mathcal{H}^2$-matrix approximation of $\mathbf{K}$ with cluster ranks $k^\tau := \#M^\tau = (m^\tau)^d$.

Remark 4.1 *Combining standard interpolation estimates with the asymptotic smoothness property* (7) *and the admissibility condition* (8)*, we get the error bound*

$$\begin{aligned}
|\kappa(x,y) - \tilde{\kappa}(x,y)| &\leq C(c_1\eta)^m \min_{(\tau,\sigma)\in P_{\mathrm{far}}} \operatorname{dist}(Q^\tau, Q^\sigma)^{-g} \\
&\leq C(c_1\eta)^m \left(\frac{\eta}{\min_{\tau\in T_I} \operatorname{diam}(Q^\tau)} \right)^g
\end{aligned}$$

for constants $C, c_1 \in \mathbb{R}_{>0}$ (cf. [3]). This means that the $\mathcal{H}^2$-matrix approximation converges exponentially in m if $c_1\eta < 1$ holds.

4.3 Variable-order interpolation

Using interpolation of constant order m, the error of the $\mathcal{H}^2$-matrix approximation is in $\mathcal{O}((c_1\eta)^m)$, while the complexity for the matrix-vector multiplication is in $\mathcal{O}(nm^d)$.

This is similar to multigrid techniques: One step of the multigrid iteration reduces the error by a fixed factor $\gamma \in]0,1[$ and requires $\mathcal{O}(n)$ operations, so m steps of the iteration will give us a precision in $\mathcal{O}(\gamma^m)$ by using $\mathcal{O}(nm)$ operations.

The number n of degrees of freedom is related to the discretization error, and we have to ensure that the error introduced by our approximation scheme or, in the multigrid context, by the iterative solution process, is of the same order of magnitude as the discretization error.

For typical discretization schemes, this means that m should be proportional to $\log(n)$, and this implies that the "true" complexity of the constant-order approximation is $\mathcal{O}(n\log^d n)$. This is clearly not optimal.

The solution is suggested by multigrid techniques: In the *nested iteration*, the multigrid iteration is performed on a coarse grid until the iterative error is proportional to the discretization error on this grid. Then the solution is interpolated on the next finer

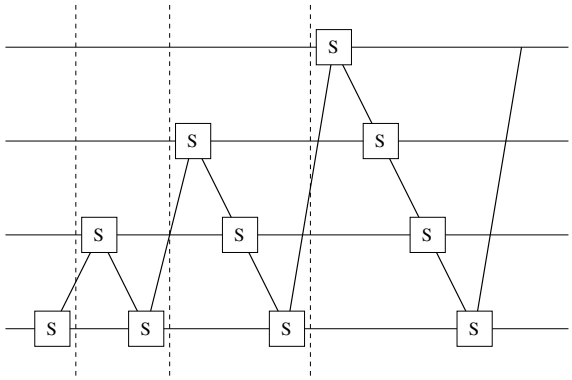


Figure 1: Multigrid iterations during a nested iteration

grid, and this interpolant is used as an initial guess for the iteration on this grid. The procedure is repeated until the finest grid has been reached, and due to the quality of the initial guesses, only a fixed number α of iterations has to be performed per level. Since the number of degrees of freedom decreases exponentially, the entire nested iteration procedure takes only $\mathcal{O}(n)$ operations to compute a sufficiently precise solution on the finest grid.

Let us consider the amount of work per level: During the computation on the coarsest grid, α multigrid steps are performed. On the next finer grid, again α multigrid steps are performed, and each of these steps involves one recursive multigrid step on the coarsest level (cf. Figure 1). Summing this up, we see that a total of $\alpha(L-\ell)+\alpha$ multigrid steps are performed on level ℓ of the hierarchy, i.e., the number of iteration steps increases linearly when the level number decreases.

In the setting of $\mathcal{H}^2$-matrices, the rank takes the place of the number of iteration steps, so the equivalent of the nested iteration will be to use a higher rank on larger clusters and a lower rank on smaller ones. This causes a problem: In the definition of our $\mathcal{H}^2$-matrix approximation of $\mathbf{K}$, we needed the inequality $m^\tau \leq m^{\tau'}$ in order to be able to prove equation (6), i.e., that the cluster bases are nested. Since this equation is crucial for our method, we look for an alternative way of defining the cluster basis.

The idea is simple: We look for approximations $\widehat{\mathcal{L}}_\nu^\tau$ of the Lagrange polynomials $\mathcal{L}_\nu^\tau$ such that the equality

$$\widehat{\mathcal{L}}_\nu^\tau = \sum_{\nu' \in M^{\tau'}} \mathbf{B}_{\nu'\nu}^{\tau',\tau} \widehat{\mathcal{L}}_{\nu'}^{\tau'}$$

holds. This equality implies (6). To this end, we introduce the piecewise interpolation operator $\widehat{\mathfrak{I}}^\tau$ by

$$\widehat{\mathfrak{I}}^\tau[u](x) := \begin{cases} u(x) & \text{if } \mathfrak{S}^\tau = \emptyset \\ \widehat{\mathfrak{I}}^{\tau'}[\mathfrak{I}^{\tau'}[u]](x) & \text{if } x \in \tau', \tau' \in \mathfrak{S}^\tau \\ 0 & \text{otherwise,} \end{cases}$$

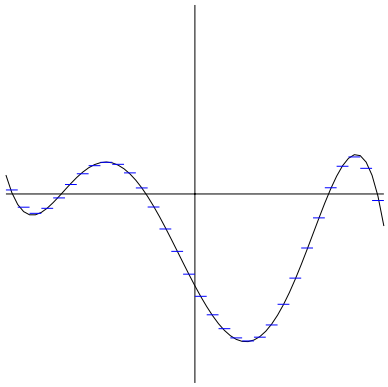


Figure 2: $\mathcal{L}_\nu^\tau$ compared to $\widehat{\mathcal{L}}_\nu^\tau$ for $m^\tau = 5$

and define the approximative Lagrange polynomials by

$$\widehat{\mathcal{L}}_\nu^\tau := \widehat{\mathfrak{I}}^\tau[\mathcal{L}_\nu^\tau].$$

We can see in Figure 2 what this means: The original Lagrange polynomials are replaced by piecewise lower-order interpolants.
By definition, we have $\widehat{\mathcal{L}}_\nu^\tau = \mathcal{L}_\nu^\tau$ for all leaves $\tau \in T_I$. For a non-leaf $\tau \in T_I$ with a son $\tau' \in \mathfrak{S}^\tau$ and for $x \in \tau'$, we find

$$\widehat{\mathcal{L}}_\nu^\tau(x) = \widehat{\mathfrak{I}}^\tau[\mathcal{L}_\nu^\tau](x) = \widehat{\mathfrak{I}}^{\tau'}[\mathfrak{I}^{\tau'}[\mathcal{L}_\nu^\tau]](x) = \sum_{\nu' \in M^{\tau'}} \mathcal{L}_\nu^\tau(x_{\nu'}^{\tau'}) \widehat{\mathfrak{I}}^{\tau'}[\mathcal{L}_{\nu'}^{\tau'}](x) = \sum_{\nu' \in M^{\tau'}} \mathbf{B}_{\nu'\nu}^{\tau',\tau} \widehat{\mathcal{L}}_{\nu'}^{\tau'}(x).$$

Using the approximative Lagrange polynomials, we can construct variable-order approximations of the kernel function $\kappa(\cdot,\cdot)$. It is possible to show that the resulting approximation error is proportional to the discretization error and that the complexity for the matrix-vector multiplication in this case is in $\mathcal{O}(n)$ [16, 5].

5 Approximation of general matrices

We have seen that $\mathcal{H}^2$-matrices are an efficient tool for the approximation of matrices resulting from the Galerkin discretization of simple integral operators. The papers [3, 5] demonstrate that our technique can be applied to other discretization schemes and integral operators involving derivatives of the kernel function.

The construction of $\mathcal{H}^2$-matrices we have presented so far requires an explicitly given kernel function and an explicitly given approximation scheme for this kernel function. There is an alternative construction that creates $\mathcal{H}^2$-matrix approximations based on purely algebraic properties of the underlying original matrix [2, 4].

We assume that a cluster tree and a block partition are given and that each index $i \in \mathcal{I}$ is contained in exactly one leaf cluster, i.e., that the leaf clusters form a partition of $\mathcal{I}$.

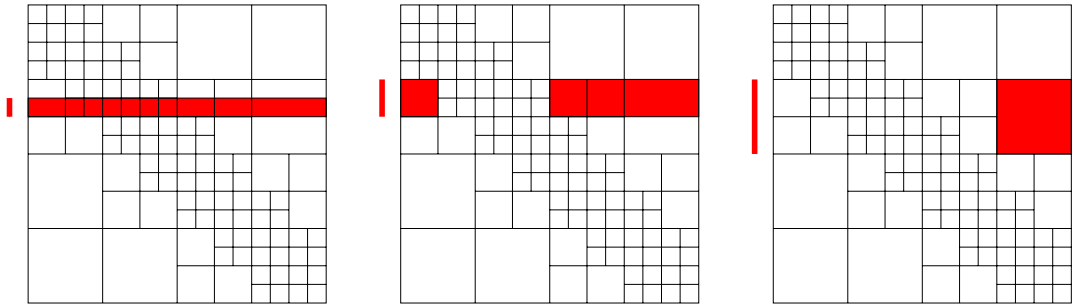


Figure 3: Matrix rows corresponding to different clusters

If we have cluster bases $(\mathbf{V}^\tau)_{\tau\in T_I}$ that are orthogonal, i.e., satisfy ${\mathbf{V}^\tau}^\top\mathbf{V}^\tau = \mathbf{I}$, the computation of the coefficient matrices $(\mathbf{S}^{\tau,\sigma})_{(\tau,\sigma)\in P_{\text{far}}}$ is straightforward: The orthogonal projection of a matrix block $\mathbf{A}|_{\tau\times\sigma}$ into the range of $\mathbf{V}^\tau$ is given by $\mathbf{V}^\tau{\mathbf{V}^\tau}^\top\mathbf{A}|_{\tau\times\sigma}$, therefore the coefficient matrices of far-field blocks are given by

$$\mathbf{S}^{\tau,\sigma} := {\mathbf{V}^\tau}^\top\mathbf{A}|_{\tau\times\sigma}\mathbf{V}^\sigma.$$

This leaves us with the task of computing an orthogonal cluster basis for each cluster $\tau\in T_I$. A cluster basis $\mathbf{V}^\tau$ will be relevant to all far-field blocks of the form $\tau^0\times\sigma$ where τ^0 is an ancestor of τ (cf. Figure 3), i.e., to the matrix block $\mathbf{A}|_{\tau\times R^\tau}$ with

$$R^\tau := \bigcup\{\sigma\in T_I \;:\; \exists\tau^0\in T_I\ (\tau^0,\sigma)\in P_{\text{far}}, \tau\subseteq\tau^0\}.$$

For leaf clusters, finding an optimal basis is simple: We compute the singular value decomposition of the matrix block $\mathbf{A}|_{\tau\times R^\tau}$ and construct the basis from the left singular values.

For non-leaf clusters $\tau\in T_I$, we have to make sure that the equation (6) is preserved. Finding a good orthogonal basis $\mathbf{V}^\tau$ for the matrix block $\mathbf{A}|_{\tau\times R^\tau}$ is equivalent to maximizing the Frobenius norm of ${\mathbf{V}^\tau}^\top\mathbf{A}|_{\tau\times R^\tau}$. If $\mathbf{V}^\tau$ satisfies (6), we have

$$\begin{aligned}{\mathbf{V}^\tau}^\top\mathbf{A}|_{\tau\times\sigma} &= \begin{pmatrix}\mathbf{B}^{\tau_1,\tau}{}^\top{\mathbf{V}^{\tau_1}}^\top & \dots & \mathbf{B}^{\tau_s,\tau}{}^\top{\mathbf{V}^{\tau_s}}^\top\end{pmatrix}\mathbf{A}|_{\tau\times R^\tau}\\ &= \begin{pmatrix}\mathbf{B}^{\tau_1,\tau}{}^\top & \dots & \mathbf{B}^{\tau_s,\tau}{}^\top\end{pmatrix}\begin{pmatrix}{\mathbf{V}^{\tau_1}}^\top\mathbf{A}|_{\tau_1\times R^\tau}\\ \vdots\\ {\mathbf{V}^{\tau_s}}^\top\mathbf{A}|_{\tau_s\times R^\tau}\end{pmatrix}\end{aligned}$$

for $\mathfrak{S}^\tau = \{\tau_1,\dots,\tau_s\}$. This means that we can compute the transfer matrices $\mathbf{B}^{\tau',\tau}$ by applying the singular value decomposition to the transformed sub-blocks. We get the following algorithm:

```
procedure CreateRowBasis(τ);
begin
  if 𝔖^τ = ∅ then
    Compute SVD of A|_{τ×R^τ}
  else begin
    𝔖^τ = {τ^1, ..., τ^s};
    for i = 1 to s do begin
      CreateRowBasis(τ_i);
      Â^{τ_i} := V^{τ_i}ᵀ A|_{τ_i×R^τ}
    end;
    Compute SVD of ( Â^{τ_1} )
                   (   ⋮    )
                   ( Â^{τ_s} )
  end
end
```

The advantage of this algorithm is that it can be applied efficiently not only to dense matrices, but also to hierarchical matrices (cf. [10, 12, 11, 1]) and even to $\mathcal{H}^2$-matrices. The latter may seem strange, but numerical examples demonstrate that the original cluster bases are not optimal and that applying the above algorithm can significantly improve the performance of the matrix-vector multiplication while reducing the storage requirements.

6 Numerical examples

6.1 Constant-order approximation in 2D

Our first experiment is to apply the $\mathcal{H}^2$-matrix approximation technique to the single layer potential $\log \|x - y\|$ on the unit circle in $\mathbb{R}^2$. We use a constant-order approach with $m = 3$ and set the admissibility parameter to $\eta = 0.8$. The results are given in Table 1.

We have discretized the integral operator by the Galerkin method based on piecewise constant basis functions. Table 1 gives the dimension of the discrete space, the time in seconds[2] for building the $\mathcal{H}^2$-matrix approximation, the time in seconds for performing one matrix-vector multiplication, the relative approximation error in the spectral norm and the memory requirements per degree of freedom.

We can see that the relative error in the Euclidean norm is almost constant and that both the runtime and the storage requirements grow linearly in the number of degrees of freedom.

[2]These times were measured using an Sun Ultra2 processor running at 248 MHz.

n	Build[s]	MV[s]	$\frac{\|\mathbf{K}-\tilde{\mathbf{K}}\|_2}{\|\mathbf{K}\|_2}$	Mem/n
1024	0.78	0.01	5.98_{-4}	1011
2048	1.49	0.03	5.98_{-4}	1014
4096	2.97	0.07	5.98_{-4}	1016
8192	6.16	0.17	5.98_{-4}	1016
16384	11.88	0.33	5.99_{-4}	1017
32768	24.25	0.66		1017
65536	48.20	1.34		1017
131072	96.63	2.75		1017
262144	194.27	5.61		1017
524288	390.56	10.91		1017

Table 1: $\mathcal{H}^2$-matrix constant-order approximation

n	Build[s]	MV[s]	$\frac{\|\mathbf{K}-\tilde{\mathbf{K}}\|_2}{\|\mathbf{K}\|_2}$	Mem/n
1024	1.05	0.02	4.84_{-4}	4171
2048	2.30	0.06	2.65_{-4}	4605
4096	4.97	0.14	1.40_{-4}	4929
8192	10.12	0.31	7.25_{-5}	5162
16384	21.01	0.62	3.71_{-5}	5324
32768	40.96	1.22	1.88_{-5}	5434
65536	83.54	2.54		5507
131072	167.79	5.16		5554
262144	338.22	10.27		5584
524288	675.67	20.81		5602

Table 2: $\mathcal{H}^2$-matrix variable-order approximation

6.2 Variable-order approximation in 2D

The relative approximation error for the constant-order approximation is constant. In typical applications, the approximation error should be of the same order as the discretization error. The way to achieve this is to use a variable-order approximation. For each cluster $\tau \in T_I$, we denote its distance to the root cluster Γ by $\mathrm{level}(\tau)$ and set $m^\tau := 1 + (L - \mathrm{level}(\tau))$, i.e., the approximation order increases by one when we pass from a cluster to its father. The results of our experiments for this algorithm are given in Table 2.

Since we are using piecewise constant basis functions, we expect the discretization error to decrease like $\mathcal{O}(n^{-1})$. We can see that the approximation error of the variable-order $\mathcal{H}^2$-matrix shows the same behaviour, so discretization and approximation error are proportional, while the computational complexity and the storage complexity are still in $\mathcal{O}(n)$.

n	Build[s]	MV[s]	$\frac{\|\mathbf{K}-\tilde{\mathbf{K}}\|_2}{\|\mathbf{K}\|_2}$	Mem/n
512	4.31	0.01	4.47_{-5}	2688
2048	24.77	0.04	4.99_{-5}	4168
8192	108.39	0.23	6.67_{-5}	4978
32768	428.68	1.00	7.24_{-5}	5282
131072	1666.90	4.01		5251
524288	6443.56	16.10		5196
2097152	25932.05	69.23		5149

Table 3: $\mathcal{H}^2$-matrix approximation with adaptive cluster bases

6.3 Adaptive approximation in 3D

As a last example, we discretize the single layer potential $1/\|x-y\|$ on the unit sphere in $\mathbb{R}^3$ by piecewise constant basis functions. We use a constant approximation order of $m = 4$ and use the algorithm from Section 5 in order to compute an orthogonalized basis.

Table 3 reports the results of the experiment[3]: The time required for building the $\mathcal{H}^2$-matrix approximation is roughly linear in the dimension of the discrete space, as is the time required for performing the matrix-vector multiplication. The same holds for the memory requirements: The ratio of needed storage and number of degrees of freedom is bounded. The relative error grows slightly, but this can be compensated by increasing the order of the interpolation if necessary.

References

[1] S. Börm, L. Grasedyck, and W. Hackbusch, *Introduction to hierarchical matrices with applications*, Tech. Rep. 18, Max Planck Institute for Mathematics in the Sciences, 2002. To appear in: Engineering Analysis with Boundary Elements.

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

[3] ——, *$\mathcal{H}^2$-matrix approximation of integral operators by interpolation*, Applied Numerical Mathematics, 43 (2002), pp. 129–143.

[4] ——, *Approximation of boundary element operators by adaptive $\mathcal{H}^2$-matrices*, Tech. Rep. 5, Max Planck Institute for Mathematics in the Sciences, 2003.

[5] S. Börm, M. Löhndorf, and J. M. Melenk, *Approximation of integral operators by variable-order interpolation*, Tech. Rep. 82, Max Planck Institute for Mathematics in the Sciences, 2002.

[3] The times where measured on a Sun Ultra3cu processor running at 900 MHz.

[6] W. Dahmen and R. Schneider, *Wavelets on manifolds I: Construction and domain decomposition*, SIAM Journal of Mathematical Analysis, 31 (1999), pp. 184–230.

[7] K. Giebermann, *Multilevel approximation of boundary integral operators*, Computing, 67 (2001), pp. 183–207.

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

[9] L. Grasedyck and W. Hackbusch, *Construction and arithmetics of $\mathcal{H}$-matrices*, Tech. Rep. 103, Max Planck Institute for Mathematics in the Sciences, 2002.

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

[11] W. Hackbusch and B. Khoromskij, *A sparse $\mathcal{H}$-matrix arithmetic: General complexity estimates*, J. Comp. Appl. Math., 125 (2000), pp. 479–501.

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

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

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

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

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