$\mathcal{H}^2$-Matrix Approximation of Integral Operators by Interpolation

Wolfgang Hackbusch and Steffen Börm

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

Abstract

Typical panel clustering methods for the fast evaluation of integral operators are based on the Taylor expansion of the kernel function and therefore usually require the user to implement the evaluation of the derivatives of this function up to an arbitrary degree.

We propose an alternative approach that replaces the Taylor expansion by simple polynomial interpolation. By applying the interpolation idea to the approximating polynomials on different levels of the cluster tree, the matrix vector multiplication can be performed in only $\mathcal{O}(np^d)$ operations for a polynomial order of p and an n-dimensional trial space.

The main advantage of our method, compared to other methods, is its simplicity: Only pointwise evaluations of the kernel and of simple polynomials have to be implemented.

1 Introduction

We consider an integral equation on a submanifold Γ of $\mathbb{R}^d$ for $d \in \mathbb{N}$, namely

Find u (in a suitable space) satisfying

$$\lambda u(x) + \int_\Gamma k(x,y)u(y)dy = f(x) \qquad (1)$$

for all $x \in \Gamma$

for a right hand side f, a parameter $\lambda \in \mathbb{R}$ and kernel function

$$k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

Preprint submitted to Elsevier Science *6 December 2001*

that has the asymptotic smoothness (cf. (18)) typical for BEM kernels.

We use a Galerkin approach for the discretization of the equation (1), i.e., we choose a family $(\Psi_i)_{i\in I}$ of suitable basis functions, set $V_h := \operatorname{span}\{\Psi_i \;:\; i\in I\}$ and get the discrete problem

Find $u_h \in V_h$ satisfying

$$\lambda \int_\Gamma u_h(x)v_h(x)dx + \int_\Gamma\int_\Gamma k(x,y)u_h(y)v_h(x)dy\; dx$$
$$= \int_\Gamma f(x)v_h(x)dx \tag{2}$$

for all $v_h \in V_h$.

We introduce matrices $M \in \mathbb{R}^{I\times I}, K \in \mathbb{R}^{I\times I}$ and vectors $\hat{u} \in \mathbb{R}^I, \hat{f} \in \mathbb{R}^I$ by setting

$$M_{ij} = \int_\Gamma \Psi_j(x)\Psi_i(x)dx, \quad \hat{f}_j = \int_\Gamma f(x)\Psi_j(x)dx,$$
$$u_h = \sum_{i\in I}\hat{u}_i\Psi_i \quad \text{and} \quad K_{ij} = \int_\Gamma\int_\Gamma k(x,y)\Psi_j(y)\Psi_i(x)dy\; dx$$

for all $i,j\in I$, which allows us to rewrite (2) in the form

$$(\lambda M + K)\hat{u} = \hat{f}. \tag{3}$$

In typical applications, the basis functions $(\Psi_i)_{i\in I}$ have local support, so the matrix M is sparse, i.e., for $n := |I|$, the construction of the matrix and its multiplication with a given vector can be accomplished in $\mathcal{O}(n)$ operations.

Since the support of the integral kernel $k(\cdot,\cdot)$ is not local, the matrix K is *not* sparse, so a straightforward computation of its entries requires at least $\mathcal{O}(n^2)$ operations, and a multiplication of a vector with this matrix will require the same amount of work.

There are different approaches for reducing the complexity: If the domain is very regular, the resulting matrix has a Toeplitz-type structure and can therefore be evaluated efficiently by performing fast Fourier transformations. If the domain is smooth, it is possible to use Wavelet compression, and this approach can be extended to piecewise smooth domains [1]. In this article, we will consider a variant of the panel clustering method [7] and the mosaic skeleton matrix approach [9] by using the framework of $\mathcal{H}^2$-matrices introduced in [6].

The idea of our variant is to replace K by a suitable approximation and thereby to reduce the complexity to $\mathcal{O}(np^d)$, where p is a parameter corresponding to the "quality" of the approximation and d, as above, is the dimension of the space in which the manifold Γ is embedded.

We will approximate the kernel function $k(\cdot,\cdot)$ by an interpolant. Since $k(\cdot,\cdot)$ will usually not be smooth on the entire domain $\Gamma \times \Gamma$, we do not use a global interpolant, but choose subsets $\Gamma_\tau \times \Gamma_\sigma$ where $k(\cdot,\cdot)$ is smooth and interpolate on each of these subsets.

If we organize the subsets Γ_τ in a hierarchical way, we end up with a multilevel block matrix where each block has a rank $\leq p^d$. The structure is that of an $\mathcal{H}$-matrix described in [3,4].

In this form, the approximating matrix requires only $\mathcal{O}(np^d \log n)$ units of storage, constructing it and multiplying it with vectors requires $\mathcal{O}(np^d \log n)$ operations.

If we exploit the fact that the kernel is locally approximated by polynomials that depend only on Γ_τ and Γ_σ and have a fixed degree, we find that the structure resembles that of $\mathcal{H}^2$-matrices described in [6]. Using an algorithm similar to the Fast Fourier Transformation, but not as limited in scope, we can reduce the storage requirements and the number of operations to $\mathcal{O}(np^d)$.

Since only quadrature and polynomial interpolation are needed in our algorithm, its implementation is very simple as compared to Wavelet compression techniques or panel clustering approaches based on Taylor expansion.

2 Cluster Tree, Block Partitioning and Types of $\mathcal{H}$-Matrices

In order to keep the selection of the subsets $\Gamma_\tau \times \Gamma_\sigma$ mentioned above algorithmically manageable, we base them on a hierarchical splitting of the index set I corresponding to the basis functions: Let $\mathcal{T}(I)$ be a binary tree and let $T(I)$ be the set of its nodes. $\mathcal{T}(I)$ is called a *binary cluster tree* if it satisfies the following conditions:

(1) $T(I) \subseteq \mathcal{P}(I)$, i.e., each node of $\mathcal{T}(I)$ is a subset of the index set I.
(2) I is the root of $\mathcal{T}(I)$.
(3) If $\tau \in T(I)$ is a leaf, then $|\tau| \leq Cp^d$, i.e., the leaves consist of a relatively small number of indices.
(4) If $\tau \in T(I)$ is *not* a leaf, then it has exactly two sons and is their disjoint union.

For each $\tau \in T(I)$, we denote the set of its sons by $S(\tau) \subseteq T(I)$.

The *support* of a cluster $\tau \in T(I)$ is given by the union of the supports of the basis functions corresponding to its elements, i.e.,

$$\Gamma_\tau := \bigcup_{i \in \tau} \Gamma_i,$$

where $\Gamma_i := \operatorname{supp} \Psi_i$ for all $i \in I$.

Next, we need an *admissibility condition* that allows us to select pairs $(\tau, \sigma) \in T(I) \times T(I)$ such that the kernel $k(\cdot, \cdot)$ is smooth enough on the domain associated with $\Gamma_\tau \times \Gamma_\sigma$.

We are going to define our interpolants on axiparallel boxes B_τ containing the set Γ_τ, so we need the kernel to be smooth on $B_\tau \times B_\sigma$ if we want a good approximation.

This can be expressed in quantitative terms by the inequality

$$\max\{\operatorname{diam}(B_\tau), \operatorname{diam}(B_\sigma)\} \leq 2\eta \operatorname{dist}(B_\tau, B_\sigma), \tag{4}$$

where $\eta \in]0, 1[$ is some parameter controlling the trade-off between the number of admissible blocks, i.e., the algorithmic complexity, and the speed of convergence, i.e., the quality of the approximation.

The condition (4) is especially suited for kernel functions with the property (18).

Figure 1 contains a cluster splitting of the unit circle. For some clusters τ, the corresponding boxes B_τ are highlighted in grey.

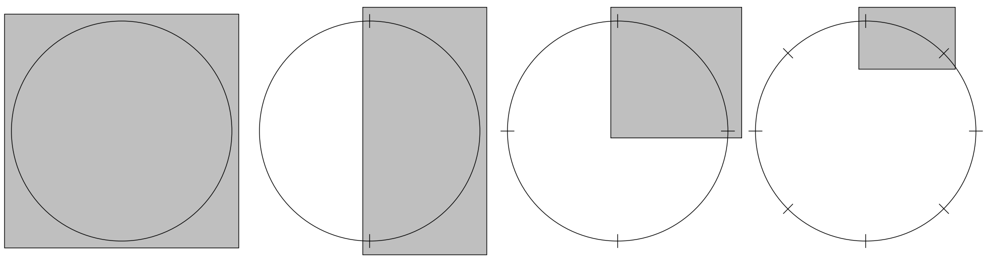

Fig. 1. Dyadic clustering of the unit circle.

The index set $I \times I$ corresponding to the matrix $K \in \mathbb{R}^{I \times I}$ is partitioned into blocks $\tau \times \sigma$ by the algorithm shown in Table 1:

Calling this procedure with $\tau = \sigma = I$ and $P = \emptyset$ creates a block partitioning of $I \times I$ consisting of admissible blocks and non-admissible blocks corresponding to leaf clusters of $\mathcal{T}(I)$.

An example of an admissible partitioning corresponding to the splitting outlined in Figure 1 can be found in Figure 2.

Table 1
BuildPartitioning algorithm

```
procedure BuildPartitioning(τ, σ, var P);
begin
  if (τ, σ) meets condition (4) then
    P := P ∪ {τ × σ}
  else
    if S(τ) ≠ ∅ and S(σ) ≠ ∅ then
      for τ' ∈ S(τ), σ' ∈ S(σ) do
        BuildPartitioning(τ', σ', P)
    else
      P := P ∪ {τ × σ}
end
```

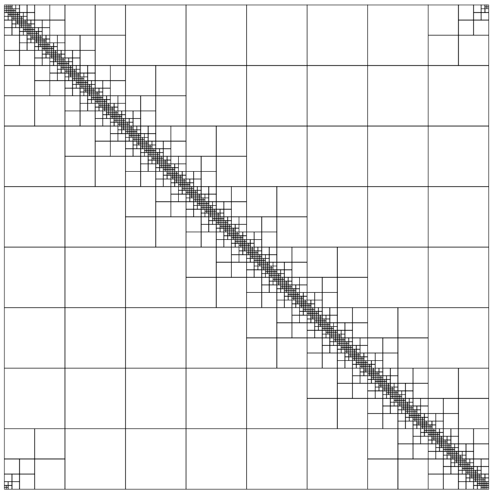

Fig. 2. Block partitioning for the unit circle with dyadic clustering.

The complexity of algorithms for the creation of suitable cluster trees and block partitionings has been analysed in detail in [2]: For typical quasi-uniform grids, a "good" cluster tree can be created in $\mathcal{O}(n \log n)$ operations, the computation of the block partitioning can be accomplished in $\mathcal{O}(n)$ operations.

Based on the cluster tree and the block partitioning, we define the following three types of data-sparse matrices:

Definition 2.1 ($\mathcal{H}$-matrices) *Let $A \in \mathbb{R}^{I \times I}$ be a matrix and P be a block partitioning of $I \times I$ consisting of admissible blocks and leaf blocks.*

(1) Let $k \in \mathbb{N}$. A is called $\mathcal{H}$-matrix of rank k*, if*

$$\operatorname{rank}(A|_{\tau \times \sigma}) \leq k \tag{5}$$

holds for each $\tau \times \sigma \in P$.

(2) A family $V = (V_\tau)_{\tau \in T(I)}$ is called a cluster basis*, if there is a $k_\tau \in \mathbb{N}$ for each $\tau \in T(I)$ such that $V_\tau \in \mathbb{R}^{\tau \times k_\tau}$.*

Let V_c, V_r *be cluster bases.* A *is called* uniform $\mathcal{H}$-matrix with respect to V_c and V_r*, if for each* $\tau \times \sigma \in P$ *there is a matrix* $S_{\tau\times\sigma}$ *satisfying*

$$A|_{\tau\times\sigma} = V_{c,\tau} S_{\tau\times\sigma} V_{r,\sigma}^T. \tag{6}$$

In this context, V_c *is called* column basis of A*,* V_r *is called* row basis of A*.*

(3) *A cluster basis* $V = (V_\tau)_{\tau\in T(I)}$ *is called* nested*, if there are transfer matrices* $B_{\tau',\tau} \in \mathbb{R}^{k_{\tau'},k_\tau}$ *for all* $\tau \in T(I)$ *with* $S(\tau) \neq \emptyset$ *and* $\tau' \in S(\tau)$ *satisfying*

$$V_\tau|_{\tau'\times k_\tau} = V_{\tau'} B_{\tau',\tau}. \tag{7}$$

A is called $\mathcal{H}^2$-matrix with respect to V_c and V_r*, if it is a uniform* $\mathcal{H}$*-matrix with respect to these cluster bases and if both bases are nested.*

3 $\mathcal{H}^2$ Construction

3.1 Approximation of the Kernel

For each cluster $\tau \in T(I)$, we fix a family $(x_\iota^\tau)_{\iota\in I_\tau}$ of interpolation points (e.g. tensor products of the zeroes of Chebyshev polynomials) in B_τ and the family $(p_\iota^\tau)_{\iota\in I_\tau}$ of corresponding Lagrange polynomials.

On a given admissible block $\tau \times \sigma \in P$, we approximate the kernel function $k(\cdot,\cdot)$ by its interpolant

$$\tilde{k}_{\tau\times\sigma}(x,y) := \sum_{\iota\in I_\tau} \sum_{\kappa\in I_\sigma} k(x_\iota^\tau, y_\kappa^\sigma) p_\iota^\tau(x) p_\kappa^\sigma(y). \tag{8}$$

3.2 Approximation of the Discrete Operator

The discrete Galerkin operator is given by

$$K_{ij} := \int_{\Gamma_i} \int_{\Gamma_j} k(x,y) \Psi_j(y) \Psi_i(x) dy\ dx$$

for finite element basis function Ψ_i, Ψ_j having their supports in Γ_i, Γ_j.

For $i \in \tau$ and $j \in \sigma$, we can replace $k(\cdot,\cdot)$ with $\tilde{k}(\cdot,\cdot)$ and find

$$\begin{aligned}\tilde{K}_{ij} &:= \int_{\Gamma_i} \int_{\Gamma_j} \tilde{k}(x,y) \Psi_i(x) \Psi_j(y) dy\ dx \\ &= \sum_{\iota\in I_\tau} \sum_{\kappa\in I_\sigma} k(x_\iota^\tau, y_\kappa^\sigma) \int_{\Gamma_i} p_\iota^\tau(x) \Psi_i(x) dx \int_{\Gamma_j} p_\kappa^\sigma(y) \Psi_j(y) dy.\end{aligned}$$

Introducing matrices $V^\tau \in \mathbb{R}^{\tau \times I_\tau}$, $V^\sigma \in \mathbb{R}^{\sigma \times I_\sigma}$ and $S^{\tau\times\sigma} \in \mathbb{R}^{I_\tau \times I_\sigma}$ with

$$V^\tau_{i\iota} := \int_{\Gamma_i} p^\tau_\iota(x)\Psi_i(x)dx, \quad V^\sigma_{j\kappa} := \int_{\Gamma_j} p^\sigma_\kappa(y)\Psi_j(y)dy, \tag{9}$$

$$S^{\tau\times\sigma}_{\iota\kappa} := k(x^\tau_\iota, y^\sigma_\kappa), \tag{10}$$

we get

$$\tilde{K}_{ij} = \left(V^\tau S^{\tau\times\sigma} V^{\sigma T}\right)_{ij}, \tag{11}$$

i.e., a representation as a uniform $\mathcal{H}$-matrix.

3.3 Nested Bases

Let Q_p be the space of polynomials spanned by tensor products of polynomials of a degree up to p. The interpolation operator is a projection onto this space, so if we use Q_p as the interpolation polynomials on *all* clusters, we can express polynomials corresponding to father clusters in terms of polynomials corresponding to son clusters *without loss*, i.e., for a cluster $\tau \in T(I)$ with a son $\tau' \in T(I)$, we find

$$p^\tau_\iota(x) = \sum_{\lambda \in I_{\tau'}} p^\tau_\iota(x^{\tau'}_\lambda) p^{\tau'}_\lambda(x). \tag{12}$$

By introducing a matrix $B^{\tau',\tau} \in \mathbb{R}^{I_{\tau'} \times I_\tau}$ by setting

$$B^{\tau',\tau}_{\lambda\iota} := p^\tau_\iota(x^{\tau'}_\lambda), \tag{13}$$

we get

$$\begin{aligned} V^\tau_{i\iota} &= \int_{\Gamma_i} p^\tau_\iota(x)\Psi_i(x)dx = \sum_{\lambda\in I_{\tau'}} p^\tau_\iota(x^{\tau'}_\lambda) \int_{\Gamma_i} p^{\tau'}_\lambda(x)\Psi_i(x)dx \\ &= \sum_{\lambda\in I_{\tau'}} B^{\tau',\tau}_{\lambda\iota} V^{\tau'}_{i\lambda} = (V^{\tau'} B^{\tau',\tau})_{i\iota} \end{aligned} \tag{14}$$

for all $i \in \tau' \subseteq \tau$, so the bases are nested in the way required for $\mathcal{H}^2$-matrices.

4 Algorithms and Complexity

We require the block partitioning P to be *sparse* in the sense of [2], i.e., that there exists a constant C_{sp} satisfying

$$\max_{\tau \in T(I)} |\{\sigma \in T(I) \ : \ \tau \times \sigma \in P\}| \leq C_{\mathrm{sp}}, \tag{15}$$

$$\max_{\sigma \in T(I)} |\{\tau \in T(I) \ : \ \tau \times \sigma \in P\}| \leq C_{\mathrm{sp}}. \tag{16}$$

Table 2

ForwardTransformation algorithm

```
procedure ForwardTransformation(σ);
begin
  if S(σ) = ∅ then
    x_σ := V^{σT} x|_σ
  else begin
    x_σ := 0;
    for σ' ∈ S(σ) do begin
      ForwardTransformation(σ');
      x_σ := x_σ + B^{σ',σ T} x_{σ'}
    end
  end
end
```

For standard situations with quasi-uniform meshes, this estimate has been established in [2].

We further assume that there is a constant C_L satisfying

$$S(\tau) = \emptyset \Rightarrow |I_\tau| \leq |\tau| \leq C_L |I_\tau| \tag{17}$$

for all $\tau \in T(I)$, i.e., the size of a leaf cluster is comparable to the number of polynomials used in the interpolation on this cluster.

Note that, due to the requirement of subsection 3.3, we have $|I_\tau| = p^d$ for all clusters $\tau \in T(I)$.

4.1 Matrix-Vector Multiplication

We split the matrix-vector multiplication $y := \tilde{K}x$ into three steps: First, the products of the input vector with the matrices $V^{\sigma T}$ are computed by recursively applying the equation (14). Then, the resulting coefficients are multiplied by $S^{\tau \times \sigma}$. In the last step, the result is transformed back from the coefficients of the cluster bases into the standard basis.

4.1.1 Forward Transformation

In this first step, we want to compute $x_\sigma := V^{\sigma T} x|_\sigma$ for all $\sigma \in T(I)$. If σ is not a leaf, i.e., if $S(\sigma) \neq \emptyset$, we have

$$V^{\sigma T} x|_\sigma = \sum_{\sigma' \in S(\sigma)} B^{\sigma',\sigma T} V^{\sigma' T} x|_{\sigma'} = \sum_{\sigma' \in S(\sigma)} B^{\sigma',\sigma T} x_{\sigma'},$$

so the algorithm shown in Table 2 can be used to compute all coefficients.

Table 3
BackwardTransformation algorithm

```
procedure BackwardTransformation(τ);
begin
  if S(τ) = ∅ then
    y|_τ := V^τ y_τ
  else
    for τ' ∈ S(τ) do begin
      y_τ' := y_τ' + B^{τ',τ} y_τ;
      BackwardTransformation(τ')
    end
end
```

Now we want to analyse the complexity of this algorithm. For each $\sigma \in T(I)$, we have $|I_\sigma| = p^d$ and find that $\mathcal{O}(p^{2d})$ operations are required for the computation of x_σ. Due to the lower bound in condition (17), there are no more than n/p^d leaves in $\mathcal{T}(I)$, so there are less than $2n/p^d$ nodes.

Therefore, the full forward transformation takes $\mathcal{O}(np^d)$ operations to complete.

4.1.2 Multiplication

In the second step, we want to compute

$$y_\tau := \sum_{\sigma \in P_\tau} S_{\tau\times\sigma} x_\sigma$$

for all $\tau \in T(I)$, where

$$P_\tau := \{\sigma \in T(I) \ : \ \tau \times \sigma \in P\}.$$

Due to the sparsity condition (15), we have $|P_\tau| \leq C_{\mathrm{sp}}$, so the computation of y_τ requires $\mathcal{O}(C_{\mathrm{sp}} p^{2d})$ operations.

The cluster tree has at most $2n/p^d$ nodes, so the second step takes $\mathcal{O}(np^d)$ operations.

4.1.3 Backward Transformation

In the last step, we transform the result from the cluster bases into the standard basis. The procedure shown in Table 3 is the adjoint of the Forward Transformation.

By the same arguments as in the case of the Forward Transformation, we find that only $\mathcal{O}(np^d)$ operations are required for the Backward Transformation, so the complete matrix-vector multiplication can be completed in $\mathcal{O}(np^d)$ operations.

4.2 Discretization

4.2.1 Basis Transformation Matrices

We first consider the computation of the matrices $B^{\tau',\tau}$. We are using tensor product interpolation, so the polynomials p_ι^τ and the corresponding interpolation points x_ι^τ can be written in the form

$$p_\iota^\tau(x_1,\dots,x_d) = p_{\iota_1}^{\tau,1}(x_1)\dots p_{\iota_d}^{\tau,d}(x_d),$$
$$x_\iota^\tau = (x_{\iota_1}^{\tau,1},\dots,x_{\iota_d}^{\tau,d})$$

for one-dimensional Lagrange polynomials $(p_j^{\tau,i})$ and corresponding interpolation points $(x_j^{\tau,i})$ on the real line.

The equation (13) takes the form

$$B_{\lambda\iota}^{\tau',\tau} = p_\iota^\tau(x_\lambda^{\tau'}) = p_{\iota_1}^{\tau,1}(x_{\lambda_1}^{\tau',1})\dots p_{\iota_d}^{\tau,d}(x_{\lambda_d}^{\tau',d}),$$

so it has the tensor product structure

$$B^{\tau',\tau} = B^{\tau',\tau,1}\otimes\dots\otimes B^{\tau',\tau,d}$$

with

$$B_{\iota_i,\lambda_i}^{\tau',\tau,i} = p_{\iota_i}^{\tau,i}(x_{\lambda_i}^{\tau',i}).$$

The computation of one of these latter matrices requires $\mathcal{O}(p^3)$ operations, the computation of all of them requires $\mathcal{O}(dp^3)$. This means that, since there are n/p^d leaves and each leaf requires $\mathcal{O}(dp^3)$ units of storage, the matrices $B^{\tau',\tau}$ can be stored in the form of tensor products in $\mathcal{O}(dnp^{3-d})$ units of storage.

Of course, we can also construct the full matrix $B^{\tau',\tau}$ from the tensor product form. This requires $\mathcal{O}(dp^3+p^{2d})$ operations and leads to a total of $\mathcal{O}(dnp^{3-d}+np^d)$ operations.

4.2.2 Leaf Basis Matrices

The computation of the matrices V^τ for leaves $\tau\in T(I)$ corresponds to the computation of the integrals

$$V_{i\iota}^\tau = \int_{\Gamma_i} p_\iota^\tau(x)\Psi_i(x)dx,$$

where Ψ_i is some finite element basis function, p_ι^τ a tensor product polynomial in Q_p and where $i \in \tau$ and $\iota \in I_\tau$. The computation of each of the $\mathcal{O}(p^{2d})$ entries using Gauss quadrature of order q requires $\mathcal{O}(p^{2d}q^d)$ operations, so all leaf basis matrices can be computed using $\mathcal{O}(np^d q^d)$ operations and stored in $\mathcal{O}(np^d)$ units of memory.

4.2.3 Admissible Blocks

According to (10), the coefficient matrices $S^{\tau\times\sigma}$ corresponding to admissible blocks are defined by

$$S_{\iota\kappa}^{\tau\times\sigma} = k(x_\iota^\tau, x_\kappa^\sigma),$$

so building one of these matrices requires $\mathcal{O}(p^{2d})$ operations. By the same arguments as before, we get a total of $\mathcal{O}(np^d)$ operations for the process of building the coefficients for all admissible blocks.

4.2.4 Non-admissible Blocks

To compute the matrices $S^{\tau\times\sigma}$ for non-admissible blocks, we have to approximate the integral

$$S_{ij}^{\tau\times\sigma} = \int_{\Gamma_i}\int_{\Gamma_j} k(x,y)\Psi_j(y)\Psi_i(x)dy\ dx.$$

As before, we use quadrature of order q and find that we need $\mathcal{O}(np^d q^d)$ operations and $\mathcal{O}(np^d)$ units of memory.

5 Error Analysis

We assume that the kernel $k(\cdot,\cdot)$ is *asymptotically smooth*, i.e., that there are constants $C_{\mathrm{as}}(n,m)$ satisfying

$$|\partial_x^\alpha \partial_y^\beta k(x,y)| \le C_{\mathrm{as}}(|\alpha|,|\beta|)\|x-y\|^{-|\alpha|-|\beta|}|k(x,y)| \tag{18}$$

for all multi-indices $\alpha,\beta \in \mathbb{N}_0^d$ and all $x,y \in \mathbb{R}^d$ with $x \neq y$.

Lemma 5.1 (Kernel approximation) *Let $\tau\times\sigma \in P$ be an admissible block and assume that* (18) *holds for $k(\cdot,\cdot)$. Then the kernel $\tilde{k}_{\tau\times\sigma}(\cdot,\cdot)$ from* (8) *satisfies the estimate*

$$\|k - \tilde{k}_{\tau\times\sigma}\|_{\infty,B_\tau\times B_\sigma} \le C_{\mathrm{apx}}(p)\eta^{p+1}\|k\|_{\infty,B_\tau\times B_\sigma} \tag{19}$$

with the constant

$$C_{\text{apx}}(p) := \frac{\sqrt{2}d(C_{\text{as}}(0,p+1)+C_{\text{as}}(p+1,0))}{(p+1)!}\frac{(C\log(p+1))^{2d-1}}{2^{p/2}}. \tag{20}$$

The behaviour of this constant will be discussed in Remark 5.2.

Proof. The interpolated kernel $\tilde{k}(\cdot,\cdot)$ can be written in the form

$$\tilde{k} = I^p_{B_\tau\times B_\sigma} k$$

by using the tensor product interpolation operator. Applying the error estimate of Lemma A.1, we find

$$\begin{aligned}
&\|k-\tilde{k}\|_{\infty,B_\tau\times B_\sigma} = \|k - I^p_{B_\tau\times B_\sigma}k\|_{\infty,B_\tau\times B_\sigma} \\
&\leq \frac{4^{-p}}{2(p+1)!}(C\log(p+1))^{2d-1}\operatorname{diam}(B_\tau\times B_\sigma)^{p+1}\sum_{j=1}^{2d}\left\|\partial_j^{p+1}k\right\|_{\infty,B_\tau\times B_\sigma}.
\end{aligned}$$

By using the admissibility condition (4), we find

$$\begin{aligned}
&\operatorname{diam}(B_\tau\times B_\sigma) = \max\{\|p_1-p_2\|_2 \ : \ p_1,p_2\in B_\tau\times B_\sigma\} \\
&= \max\{\|(x_1,y_1)-(x_2,y_2)\|_2 \ : \ (x_1,y_1),(x_2,y_2)\in B_\tau\times B_\sigma\} \\
&= \sqrt{\max\{\|(x_1,y_1)-(x_2,y_2)\|_2^2 \ : \ x_1,x_2\in B_\tau, \ y_1,y_2\in B_\sigma\}} \\
&= \sqrt{\max\{\|x_1-x_2\|_2^2+\|y_1-y_2\|_2^2 \ : \ x_1,x_2\in B_\tau, \ y_1,y_2\in B_\sigma\}} \\
&\leq \sqrt{\operatorname{diam}(B_\tau)^2+\operatorname{diam}(B_\sigma)^2} \leq \sqrt{2}\max\{\operatorname{diam}(B_\tau),\operatorname{diam}(B_\sigma)\} \\
&\leq 2\eta\sqrt{2}\operatorname{dist}(B_\tau,B_\sigma),
\end{aligned}$$

and therefore

$$\begin{aligned}
&\|k-\tilde{k}\|_{\infty,B_\tau\times B_\sigma} \\
&\leq \frac{\sqrt{2}}{(p+1)!}\frac{(C\log(p+1))^{2d-1}}{2^{p/2}}\eta^{p+1}\operatorname{dist}(B_\tau,B_\sigma)^{p+1}\sum_{j=1}^{2d}\left\|\partial_j^{p+1}k\right\|_{\infty,B_\tau\times B_\sigma}.
\end{aligned}$$

The partial derivatives can be estimated by using the asymptotic smoothness (18) as follows:

$$\begin{aligned}
|\partial_j^{p+1}k(x,y)| &\leq C_{\text{as}}(p+1,0)\|x-y\|^{-(p+1)}|k(x,y)| \\
&\leq C_{\text{as}}(p+1,0)\operatorname{dist}(B_\tau,B_\sigma)^{-(p+1)}|k(x,y)|
\end{aligned}$$

for $j\leq d$ and

$$|\partial_j^{p+1}k(x,y)| \leq C_{\text{as}}(0,p+1)\operatorname{dist}(B_\tau,B_\sigma)^{-(p+1)}|k(x,y)|$$

for $j > d$, so we can use

$$\sum_{j=1}^{2d} \left\| \partial_j^{p+1} k \right\|_{\infty, B_\tau \times B_\sigma}$$
$$\leq d(C_{\mathrm{as}}(p+1,0) + C_{\mathrm{as}}(0,p+1)) \operatorname{dist}(B_\tau, B_\sigma)^{-(p+1)} \|k\|_{\infty, B_\tau \times B_\sigma}$$

to get the desired result

$$\begin{aligned}
&\|k - \tilde{k}\|_{\infty, B_\tau \times B_\sigma} \\
&\leq \frac{\sqrt{2}}{(p+1)!} \frac{(C \log(p+1))^{2d-1}}{2^{p/2}} \eta^{p+1} \operatorname{dist}(B_\tau, B_\sigma)^{p+1} \sum_{j=1}^{2d} \left\| \partial_j^{p+1} k \right\|_{\infty, B_\tau \times B_\sigma} \\
&\leq \frac{\sqrt{2}}{(p+1)!} \frac{(C \log(p+1))^{d-1}}{2^{p/2}} \eta^{p+1} \frac{\operatorname{dist}(B_\tau, B_\sigma)^{p+1}}{\operatorname{dist}(B_\tau, B_\sigma)^{-(p+1)}} \\
&\qquad (d\ C_{\mathrm{as}}(0,p+1) + d\ C_{\mathrm{as}}(p+1,0))\ \|k\|_{B_\tau \times B_\sigma} \\
&\leq C_{\mathrm{apx}}(p) \eta^{p+1} \frac{\operatorname{dist}(B_\tau, B_\sigma)^{p+1}}{\operatorname{dist}(B_\tau, B_\sigma)^{p+1}} \|k\|_{B_\tau \times B_\sigma} \\
&\leq C_{\mathrm{apx}}(p) \eta^{p+1} \|k\|_{B_\tau \times B_\sigma}.
\end{aligned}$$

■

Remark 5.2 (Convergence) *For typical kernels, an estimate of the type*

$$C_{\mathrm{as}}(n,m) \leq c_1 c_0^{n+m} (n+m)!$$

holds, leading to

$$C_{\mathrm{apx}}(p) \leq 2\sqrt{2} c_1 d \frac{(C \log(p+1))^{2d-1}}{2^{p/2}} c_0^{p+1}.$$

If $c_0 \eta < 1$, the approximation $\tilde{k}(\cdot,\cdot)$ will converge to $k(\cdot,\cdot)$ faster than $(c_0\eta)^{p+1}$.

If the continuous operator is W-elliptic for some Hilbert space W with $W_h := \operatorname{span}\{\Psi_i \ : \ i \in I\} \subseteq W$, then we can proceed as in [5, Lemma 3] to prove the estimate

$$\begin{aligned}
&\|u - \tilde{u}_h\|_W \\
&\leq c \left(\inf_{v_h \in W_h} \|u - v_h\|_W + C_{\mathrm{apx}}(p) \eta^{p+1} \|\tilde{K}\|_{W_h \to W_h'} \|u\|_W \right).
\end{aligned}$$

6 Extensions

Our construction can be extended to other integral operators and other discretization techniques by simple modification of the definition of V^τ:

6.1 Collocation

If we want to discretize by the collocation method, we have an additional family (ξ_i) of collocation points and the discrete operator is given by

$$K_{ij} := \int_{\Gamma_j} k(\xi_i, y)\Psi_j(y)dy.$$

Replacing once more $k(\cdot,\cdot)$ by $\tilde{k}(\cdot,\cdot)$, we find

$$\begin{aligned}\tilde{K}_{ij} &:= \int_{\Gamma_j} \tilde{k}(\xi_i, y)\Psi_j(y)dy \\ &= \sum_{\iota\in I_\tau}\sum_{\kappa\in I_\sigma} k(x_\iota^\tau, y_\kappa^\sigma)p_\iota^\tau(\xi_i)\int_{\Gamma_j} p_\kappa^\sigma(y)\Psi_j(y),\end{aligned}$$

so we can define $V^{\tau,\mathrm{coll}} \in \mathbb{R}^{\tau\times I_\tau}$ by setting

$$V_{i\iota}^{\tau,\mathrm{coll}} := p_\iota^\tau(\xi_i)$$

and get

$$\tilde{K}_{ij} = \left(V^{\tau,\mathrm{coll}}S^{\tau\times\sigma}V^{\sigma T}\right)_{ij},$$

i.e., the same result as in (11) with only V^τ replaced by $V^{\tau,\mathrm{coll}}$. As before, $V^{\tau,\mathrm{coll}}$ can be expressed in terms of the matrices $V^{\tau',\mathrm{coll}}$ corresponding to the sons τ' of τ.

6.2 Double Layer Potential

If we want to discretize the double layer potential (using the Galerkin approach again), we have to compute

$$K_{ij} := \int_{\Gamma_i}\int_{\Gamma_j}\left(\frac{\partial}{\partial n_y}k(x,y)\right)\Psi_j(y)\Psi_i(x)dy\ dx.$$

Replacing $k(\cdot,\cdot)$ by its approximation $\tilde{k}(\cdot,\cdot)$, we get

$$\begin{aligned}\tilde{K}_{ij} &:= \int_{\Gamma_i}\int_{\Gamma_j}\left(\frac{\partial}{\partial n_y}\tilde{k}(x,y)\right)\Psi_j(y)\Psi_i(x)dy\ dx \\ &= \sum_{\iota\in I_\tau}\sum_{\kappa\in I_\sigma} k(x_\iota^\tau, y_\kappa^\sigma)\int_{\Gamma_i} p_\iota^\tau(x)\Psi_i(x)dx\int_{\Gamma_j}\left(\frac{\partial}{\partial n_y}p_\kappa^\sigma(y)\right)\Psi_j(y)dy,\end{aligned}$$

so by defining $V^{\sigma,\mathrm{DLP}} \in \mathbb{R}^{\sigma\times I_\sigma}$ as

$$V_{i\iota}^{\sigma,\mathrm{DLP}} := \int_{\Gamma_j}\left(\frac{\partial}{\partial n_y}p_\kappa^\sigma(y)\right)\Psi_j(y)dy,$$

we once more get the representation

$$\tilde{K}_{ij} = \left(V^\tau S^{\tau\times\sigma} V^{\sigma,\mathrm{DLP}^T} \right)_{ij},$$

similar to (11).

This means that for the typical variants of the integral operator (single layer potential, double layer potential and its transposed, hypersingular operator) and for the typical discretization methods (collocation and Galerkin) the matrices $S^{\tau\times\sigma}$ and $B^{\tau',\tau}$ stay the same, only the matrices V^τ corresponding to the leaves of the cluster tree have to be modified.

The construction leads directly to $\mathcal{H}^2$-matrices, so if tensor products of polynomials of a degree up to p are used for a kernel function defined in $\mathbb{R}^d$, the rank will be $k = p^d$ and the complexity of the matrix-vector multiplication will be $\mathcal{O}(nk) = \mathcal{O}(np^d)$.

6.3 Variable Order Approximation

In [8], a variation of the $\mathcal{H}^2$-technique is investigated that varies the polynomial order corresponding to the size of the clusters: For small clusters, a lower order approximation is sufficient, while larger clusters require a higher order.

The same approach can be used in the context of our method: If the polynomial order corresponding to a son cluster τ' is lower than that of its father τ, the equation (12) yields only an approximation

$$p_\iota^\tau(x) \approx \tilde{p}_\iota^\tau(x) := \sum_{\lambda\in I_{\tau'}} p_\iota^\tau(x_\lambda^{\tau'}) p_\lambda^{\tau'}(x)$$

of the possibly higher-order polynomial p_ι^τ by lower-order polynomials $p_\lambda^{\tau'}$.

If the growth of the polynomial order is "slow enough", e.g., only polynomial with respect to the level of the clusters, then the storage requirements and the complexity of the discretization and the matrix-vector multiplication can be reduced to $\mathcal{O}(n)$, i.e, to optimal complexity.

7 Numerical Experiments

We applied our technique to a simple model problem: The discretization of the single layer potential on the unit circle by a Galerkin approach using a

Table 4
Unit circle, single layer potential, $\eta = 0.8$, $p(\ell) = 1 + (\ell_{\max} - \ell)$.

n	Build/s	MVM/s	$\|A - \tilde{A}\|_2/\|A\|_2$	Memory per DoF
1024	1.05	0.02	0.000483583	4171
2048	2.30	0.06	0.00026483	4605
4096	4.97	0.14	0.000140073	4929
8192	10.12	0.31	0.0000725354	5162
16384	21.01	0.62	0.0000370742	5324
32768	40.96	1.22	0.0000187955	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

regular grid of piecewise constant functions. The results are reported in Table 4.

We can see the linear growth of the time for the construction of the matrix and for the matrix vector multiplication. The relative approximation error appears to be proportional to the approximation error. The memory requirements per degree of freedom are bounded.

A Multidimensional Interpolation Estimate

We denote the Chebyshev points in the interval $[-1, 1]$ by $(\xi_i)_{i=0}^p$, i.e., we have

$$\xi_i = \cos\left(\frac{2i+1}{2p+2}\pi\right),$$

and the corresponding Lagrange polynomials by $(p_i)_{i=0}^p$. The one-dimensional interpolation operator is given by

$$\tilde{I}^p : C[-1,1] \to \mathcal{P}^p, \quad u \mapsto \sum_{i=0}^p u(\xi_i) p_i.$$

This operator satisfies the error estimate

$$\|\tilde{I}^p u - u\|_{\infty,[-1,1]} \leq \frac{2^{-p}}{(p+1)!} \|u^{(p+1)}\|_{\infty,[-1,1]} \tag{A.1}$$

and the stability estimate

$$\|\tilde{I}^p u\|_{\infty,[-1,1]} \leq C \log(p+1) \|u\|_{\infty,[-1,1]}. \tag{A.2}$$

We are interested in a result for the d-dimensional tensor product interpolation operator I_B^p on a box

$$B := \prod_{j=1}^{d} [a_j, b_j].$$

The interpolation error can be bounded as follows:

Lemma A.1 (Tensor product interpolation) *We have*

$$\begin{aligned}
&\|I_{B_d}^p u - u\|_{\infty,B_d} \\
&\leq \frac{2^{-p}}{(p+1)!} \sum_{k=1}^{d} (C \log(p+1))^{k-1} \left(\frac{b_j - a_j}{2}\right)^{p+1} \left\|\partial_j^{p+1} u\right\|_{\infty,B_d} \qquad (A.3)\\
&\leq \frac{4^{-p}}{2(p+1)!} (C \log(p+1))^{d-1} \operatorname{diam}(B_d)^{p+1} \sum_{j=1}^{d} \left\|\partial_j^{p+1} u\right\|_{\infty,B_d}
\end{aligned}$$

for $u \in C^{p+1}(B)$.

Proof. We denote by

$$\Phi_j : [-1,1] \to [a_j, b_j], \quad t \mapsto \frac{1}{2}\left((1+t)b_j + (1-t)a_j\right)$$

the transformation from the unit interval to $[a_j, b_j]$.

For each $j \in \{1, \ldots, d\}$, we introduce the operator

$$\begin{aligned}
I_j^p :& C(B) \to C(B), \\
u \mapsto& \left(x \mapsto \sum_{i=0}^{p} u(x_1, \ldots, x_{j-1}, \Phi_j(\xi_i), x_{j+1}, \ldots, x_d)\, p_i(\Phi_j^{-1}(x_j))\right)
\end{aligned}$$

interpolating functions in the j-th coordinate direction only.

The estimates (A.1) and (A.2) can be applied to I_j^p and take the form

$$\|I_j^p u - u\|_{\infty,B} \leq \frac{2^{-p}}{(p+1)!} \left(\frac{b_j - a_j}{2}\right)^{p+1} \|\partial_j^{p+1} u\|_{\infty,B}, \tag{A.4}$$

$$\|I_j^p u\|_{\infty,B} \leq (C \log(p+1)) \|u\|_{\infty,B}. \tag{A.5}$$

The tensor product interpolation operator can be written as

$$I_B^p = I_1^p \ldots I_d^p = \prod_{j=1}^{d} I_j^p.$$

Due to this product representation, we can derive the following estimate:

$$
\begin{aligned}
\|I_B^p u - u\|_{\infty,B} &= \left\| \prod_{j=1}^{d} I_j^p u - u \right\|_{\infty,B} \\
&= \left\| \prod_{j=1}^{d} I_j^p u + \sum_{k=1}^{d-1} \left(\prod_{j=1}^{k} I_j^p u - \prod_{j=1}^{k} I_j^p u \right) - u \right\|_{\infty,B} \\
&= \left\| \sum_{k=1}^{d} \left(\prod_{j=1}^{k} I_j^p u - \prod_{j=1}^{k-1} I_j^p u \right) \right\|_{\infty,B} \leq \sum_{k=1}^{d} \left\| \left(\prod_{j=1}^{k-1} I_j^p \right) (I_k^p u - u) \right\|_{\infty,B} \\
&\overset{(A.5)}{\leq} \sum_{k=1}^{d} \left(\prod_{j=1}^{k-1} (C \log(p+1)) \right) \|I_k^p u - u\|_{\infty,B} \\
&\overset{(A.4)}{\leq} \sum_{k=1}^{d} (C \log(p+1))^{k-1} \frac{2^{-p}}{(p+1)!} \left(\frac{b_k - a_k}{2} \right)^{p+1} \|\partial_k^{p+1} u\|_{\infty,B}.
\end{aligned}
$$

■

References

[1] W. Dahmen and R. Schneider, *Wavelets on manifolds I: Construction and domain decomposition*, SIAM J. of Math. Anal., (1999), pp. 184–230.

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

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

[4] W. Hackbusch and B. Khoromskij, *A sparse matrix arithmetic based on $\mathcal{H}$-Matrices. Part II: Application to multi-dimensional problems*, Computing, 64 (2000), pp. 21–47.

[5] ——, *A sparse $\mathcal{H}$-matrix arithmetic: General complexity estimates*, J. Comp. Appl. Math., (2000), pp. 479–501.

[6] 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., 2000, pp. 9–29.

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

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

[9] E. Tyrtyshnikov, *Mosaic-skeleton approximation*, Calcolo, (1996), pp. 47–57.