Max-Planck-Institut
für Mathematik
in den Naturwissenschaften
Leipzig

Approximation of boundary element operators by adaptive $\mathcal{H}^2$-matrices

(revised version: April 2004)

by

Steffen Börm and Wolfgang Hackbusch

Preprint no.: 5 2003

Approximation of boundary element operators by adaptive $\mathcal{H}^2$-matrices

Steffen Börm, Wolfgang Hackbusch

April 22, 2004

Abstract

The discretization of integral operators corresponding to non-local kernel functions typically gives rise to densely populated matrices. In order to be able to treat these matrices in an efficient manner, they have to be compressed, e.g., by panel clustering algorithms, multipole expansions or wavelet techniques.

By choosing the correct panel clustering approach, the resulting approximation of the matrix can be written in the form of a so-called $\mathcal{H}^2$-matrix. The $\mathcal{H}^2$-matrix representation can be computed for fairly general kernel functions by a black box algorithm that requires only pointwise evaluations of the kernel function.

Although this technique leads to good results, the expansion system tends to contain a certain level of redundancy that leads to an unnecessarily high complexity for the memory requirements and the matrix-vector multiplication. We present two variants of the original method that can compress the matrix even further. Both methods work on the fly, i.e., it is not necessary to keep the original $\mathcal{H}^2$-matrix in memory, and both methods perform an algebraic compression, so that the black box character of the algorithm is preserved.

1 Introduction

1.1 Model Problem

We consider integral operators of the form

$$K[u](x) = \int_\Gamma \kappa(x,y)u(y)\,dy \qquad (x \in \Gamma), \tag{1.1}$$

where u is a suitable function defined on the boundary Γ of a domain $\Omega \subseteq \mathbb{R}^d$, where $\kappa(\cdot,\cdot)$ is a kernel function defined on $\mathbb{R}^d \times \mathbb{R}^d$, possibly with a singularity at the diagonal $\{(x,x) : x \in \mathbb{R}^d\}$. Discretizing this operator by a Galerkin method leads to a matrix $\mathbf{K} \in \mathbb{R}^{\mathfrak{I}\times\mathfrak{I}}$ defined by

$$\mathbf{K}_{ij} := \int_\Gamma \int_\Gamma \kappa(x,y)\Phi_j(y)\Phi_i(x)\,dy\,dx \qquad (i,j \in \mathfrak{I}), \tag{1.2}$$

where $(\Phi_i)_{i\in\mathfrak{I}}$ is the set of basis functions. Typical kernel function $\kappa(\cdot,\cdot)$ (e.g., from BEM applications) have non-local support, therefore the matrix $\mathbf{K}$ will be densely populated. If we store $\mathbf{K}$ in the typical two-dimensional array, we will need N^2 units of memory ($N := \#\mathfrak{I}$), and obviously this complexity is not acceptable for high-dimensional problems.

1.2 Compression Techniques

There are different techniques for reducing the complexity: We can use the fast Fourier transform to diagonalize $\mathbf{K}$ if this matrix has circulant Toeplitz structure. This leads to a complexity of $\mathcal{O}(N \log N)$, but the Toeplitz structure occurs only in special situations.

Due to these restrictions, more general techniques have been developed that replace the matrix $\mathbf{K}$ by data-sparse approximations $\tilde{\mathbf{K}}$ having a complexity of $\mathcal{O}(N \log^\lambda N)$ (where λ depends on the choice of the method): The panel clustering method [10] replaces the kernel function locally by separable functions, e.g., Taylor expansions or polynomial interpolants. The multipole approach [11, 6] uses a more sophisticated expansion of the kernel function that is more efficient, but requires the analytical expansion formulae for each kernel under consideration. Finally, we can use the wavelet compression technique [5] provided we are willing to use wavelet discretizations and can construct suitable wavelet spaces for the domain Γ.

The main advantage of the panel clustering technique is that it is relatively simple, that it can be applied to a large number of practical problems and that it can be implemented as a black box method. The main disadvantage is that the standard implementations are based on an approximation of the kernel function in d-dimensional subdomains, even if Γ is only $(d-1)$-dimensional.

If we want to perform not only matrix-vector multiplications, but also matrix-matrix additions, multiplications or even the inversion of matrices efficiently, most of the techniques mentioned before can not be applied directly. The method of hierarchical matrices ($\mathcal{H}$-matrices) [7, 8, 1] generalizes the concept of separable expansions in order to find a representation of matrices that makes it possible to perform the sophisticated arithmetic operations mentioned above with complexity $\mathcal{O}(N \log^{\lambda} N)$, where λ depends on the type of operation. $\mathcal{H}^2$-matrices [9, 3, 4] are a refinement of $\mathcal{H}$-matrices that introduce an additional hierarchical structure in order to reach the *optimal complexity* $\mathcal{O}(N)$ for the matrix-vector multiplication.

1.3 Adaptive Panel Clustering Method

Our goal is to find a panel clustering algorithm that "automatically" finds improved expansion systems without sacrificing the black box character and general applicability of the original method.

The idea is to start with a representation of the boundary element matrix by an $\mathcal{H}^2$-matrix constructed by means of polynomial interpolation of the kernel function and then apply an algebraic optimization that removes superfluous functions from the expansion system while controlling the approximation error. Due to this additional optimization, setting up the optimized $\mathcal{H}^2$-matrix has a higher complexity than in the standard case, but the complexity of the matrix-vector multiplication is reduced significantly.

Another advantage of using standard interpolation as the basis of our method is that the total approximation error of the adaptive method is a combination of the well-known interpolation error and the user-defined error bound of the algebraic optimization procedure.

1.4 Organization of this Paper

This paper is organized in five sections: In the current section, we will introduce a model problem and discuss some of the related algorithms. The next section is devoted to the definition of the $\mathcal{H}^2$-approximation of the matrix corresponding to the model problem. The third section describes two techniques for compressing the original $\mathcal{H}^2$-approximation, and the fourth section contains numerical experiments that demonstrate that the compression rates achieved by our techniques are much better than those of standard methods.

2 Approximation of integral operators by $\mathcal{H}^2$-matrices

In this section, we give a short introduction to $\mathcal{H}^2$-matrices and a simple method for using them to approximate matrices corresponding to integral operators (cf. [3]).

2.1 Interpolation

Due to reasons that will become clear later on, we cannot hope to find a global approximation of the matrix $\mathbf{K}$. Therefore we consider only a submatrix corresponding to a block $\tau \times \sigma$, where $\tau, \sigma \subseteq \mathfrak{I}$ (recall that $\mathfrak{I}$ is the index set corresponding to the finite element space). Let $B^\tau, B^\sigma \subseteq \mathbb{R}^d$ be axis-parallel boxes satisfying

$$\operatorname{supp} \Phi_i \subseteq B^\tau, \quad \operatorname{supp} \Phi_j \subseteq B^\sigma \qquad (i \in \tau, j \in \sigma).$$

The boxes B^τ and B^σ will be called the *bounding boxes* corresponding to τ and σ.

Now we need to find a separable approximation of the kernel function $\kappa(\cdot,\cdot)$ on $B^\tau \times B^\sigma$. The simplest possible approach is to use interpolation: We fix an m-th order d-dimensional interpolation operator

$$\mathcal{I} : C([-1,1]^d) \to \mathcal{Q}^{d,m},$$

where $\mathcal{Q}^{d,m}$ denotes the set of d-dimensional polynomials of order m. For any d-dimensional axis-parallel box B, we introduce the transformed interpolation operator

$$\mathcal{I}^B : C(B) \to \mathcal{Q}^{d,m}, \quad u \mapsto (\mathcal{I}[u \circ \Psi]) \circ \Psi^{-1},$$

where Ψ is the standard affine mapping from $[-1,1]^d$ to B.

The approximation of $\kappa(\cdot,\cdot)$ on the domain $B^\tau \times B^\sigma$ is given by

$$\tilde{\kappa}^{\tau,\sigma} := (\mathcal{I}^{B^\tau} \otimes \mathcal{I}^{B^\sigma})\kappa \in \mathcal{Q}^{2d,m}.$$

If we denote the interpolation points corresponding to $\mathcal{I}^{B^\tau}$ and $\mathcal{I}^{B^\sigma}$ by $(x^\tau_\nu)_{\nu\in K}$ and $(x^\sigma_\mu)_{\mu\in K}$ and the corresponding Lagrange polynomials by $(\mathcal{L}^\tau_\nu)_{\nu\in K}$ and $(\mathcal{L}^\sigma_\mu)_{\mu\in K}$, we have

$$\mathcal{I}^{B^\tau} u = \sum_{\nu\in K} u(x^\tau_\nu)\mathcal{L}^\tau_\nu, \qquad \mathcal{I}^{B^\sigma} v = \sum_{\mu\in K} v(x^\sigma_\mu)\mathcal{L}^\sigma_\mu$$

and therefore

$$\tilde{\kappa}^{\tau,\sigma}(x,y) = ((\mathcal{I}^{B^\tau} \otimes \mathcal{I}^{B^\sigma})\kappa)(x,y) = \sum_{\nu\in K}\sum_{\mu\in K} \kappa(x^\tau_\nu, x^\sigma_\mu)\mathcal{L}^\tau_\nu(x)\mathcal{L}^\sigma_\mu(y),$$

i.e., in $\tilde{\kappa}^{\tau,\sigma}(\cdot,\cdot)$, we have found a separable approximation of $\kappa(\cdot,\cdot)$. Replacing $\kappa(x,y)$ by $\tilde{\kappa}^{\tau,\sigma}(x,y)$ in equation (1.2) leads to the approximated matrix entries

$$\begin{aligned}\tilde{\mathbf{K}}_{ij} &:= \int_\Gamma\int_\Gamma \tilde{\kappa}^{1,2}(x,y)\Phi_j(y)\Phi_i(x)\,dy\,dx \\ &= \sum_{\nu\in K}\sum_{\mu\in K} \kappa(x^1_\nu, x^2_\mu)\int_\Gamma \mathcal{L}^1_\nu(x)\Phi_i(x)\,dx \int_\Gamma \mathcal{L}^2_\mu(y)\Phi_j(y)\,dy = (\mathbf{V}^1\mathbf{S}^{1,2}\mathbf{V}^{2^\top})_{ij} \qquad (2.1)\end{aligned}$$

for $i \in \tau$ and $j \in \sigma$, where

$$\mathbf{V}^1_{i\nu} := \int_\Gamma \mathcal{L}^1_\nu(x)\Phi_i(x), \quad \mathbf{V}^2_{j\mu} := \int_\Gamma \mathcal{L}^2_\mu(y)\Phi_j(y) \quad \text{and} \quad \mathbf{S}^{1,2}_{\nu\mu} := \kappa(x^1_\nu, x^2_\mu).$$

This factorized form can be evaluated in $\mathcal{O}(k\#\tau + k\#\sigma + k^2)$ operations for $k := \#K$ and is therefore much more efficient than the standard form if k is significantly smaller than N.

2.2 Approximation Error

Let us now take a look at the error introduced by replacing κ by its approximation $\tilde{\kappa}^{1,2}$. For tensor product interpolation, error estimates of the form

$$\|\kappa - \tilde{\kappa}^{1,2}\|_{\infty,B^1\times B^2} \le C_{\text{in}}(m)\frac{c_1^m}{(m+1)!}\operatorname{diam}(B^1\times B^2)^{m+1}\sum_{p=1}^{2d}\|\partial_p^{m+1}\kappa\|_{\infty,B^1\times B^2} \qquad (2.2)$$

hold, where $c_1 \in \mathbb{R}_{>0}$ is a constant and $C_{\text{in}}(m)$ is a polynomial in m (cf. [3]). In order to make use of this estimate, we need a bound for the derivatives of κ. Typical kernel functions are *asymptotically smooth*, i.e., they have a singularity of order $g \in \mathbb{N}_0$ at $x = y$ and satisfy the inequality

$$|\partial_x^\alpha \partial_y^\beta \kappa(x,y)| \le C_{\text{apx}}(\alpha+\beta)!(c_0\|x-y\|)^{-g-|\alpha|-|\beta|} \qquad (2.3)$$

for some constants $C_{\text{apx}}, c_0 \in \mathbb{R}_{>0}$. Combining this inequality with (2.2) , we find

$$\|\kappa - \tilde{\kappa}^{\tau,\sigma}\|_{\infty,B^\tau\times B^\sigma} \le C_{\text{in}}(m)C_{\text{apx}}\left(\frac{c_1\operatorname{diam}(B^\tau\times B^\sigma)}{c_0\operatorname{dist}(B^\tau,B^\sigma)}\right)^{m+1}\operatorname{dist}(B^\tau,B^\sigma)^{-g}.$$

This estimate implies that we can expect good convergence only if the diameter of $B^\tau \times B^\sigma$ can be bounded by the distance of the boxes, i.e., if B^τ and B^σ satisfy an *admissibility condition* of the type

$$\operatorname{diam}(B^\tau\times B^\sigma) \le \eta\operatorname{dist}(B^\tau,B^\sigma) \qquad (2.4)$$

for some parameter $\eta \in \mathbb{R}_{>0}$. If this inequality holds, we find

$$\|\kappa - \tilde{\kappa}^{\tau,\sigma}\|_{\infty, B^\tau \times B^\sigma} \leq C_{\mathrm{in}}(m) C_{\mathrm{apx}} \left(\frac{c_1 \eta}{c_0} \right)^{m+1} \operatorname{dist}(B^\tau, B^\sigma)^{-g}, \tag{2.5}$$

i.e., we have exponential convergence in m if $\eta < c_0/c_1$ holds.

Remark 2.1 *The Newton kernel $\kappa(x,y) = 1/\|x-y\|$ satisfies (2.3) with $c_0 = 1$. Tensor product Chebyshev interpolation satisfies (2.2) with $c_1 = 1/4$. This implies that $\eta \in (0,4)$ guarantees exponential convergence of the kernel approximation.*

2.3 Local Approximation

The admissibility condition (2.4) implies $\operatorname{dist}(B^\tau, B^\sigma) > 0$, so we cannot expect to find a global approximation of the form (2.1) for the entire matrix. Instead, we split the matrix into suitable blocks and approximate each block separately.

In order to get an efficient algorithm, we use a hierarchical approach to construct the block decomposition: We organize the degrees of freedom in the form of a *cluster tree*, i.e., a tree with root $\mathfrak{I}$ and the property that if a node $\tau \subseteq \mathfrak{I}$ is not a leaf, it is the disjoint union of all of its sons. The nodes of a cluster tree will be called *clusters*.

For a given cluster tree $\mathcal{T}_{\mathfrak{I}}$, we denote the set of clusters by $T_{\mathfrak{I}}$ and the set of sons for a given cluster $\tau \in T_{\mathfrak{I}}$ by *sons*(τ). We fix a bounding box B^τ for each cluster $\tau \in T_{\mathfrak{I}}$.

We can use the admissibility condition (2.4) in combination with the cluster tree to construct the desired partition of the matrix: A pair (τ, σ) is called *admissible*, if condition (2.4) holds. We start with $(\mathfrak{I}, \mathfrak{I})$ and split a cluster pair as long as it is not admissible. This leads to the following algorithm:

```
procedure subdivide($\tau$, $\sigma$, var $P$);
begin if $(\tau,\sigma)$ is admissible then $P := P \cup \{(\tau,\sigma)\}$
      else if $sons(\tau) = \emptyset$ or $sons(\sigma) = \emptyset$ then $P := P \cup \{(\tau,\sigma)\}$
           else  for all $\tau' \in sons(\tau)$ do
                   for all $\sigma' \in sons(\sigma)$ do subdivide($\tau'$, $\sigma'$, $P$)
end;
```

```
procedure divide(var $P$);
begin $P := \emptyset$; subdivide($\mathcal{I}$, $\mathcal{I}$, $P$) end;
```

This algorithm gives us a set P satisfying

$$\mathfrak{I} \times \mathfrak{I} = \bigcup_{(\tau,\sigma) \in P} \tau \times \sigma,$$

i.e., a partition of the index set $\mathfrak{I} \times \mathfrak{I}$ corresponding to the matrix $\mathbf{K}$. Obviously, an entry (τ, σ) can only appear in P if either it is admissible or if τ or σ is a leaf. This distinction is represented by the splitting

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

of P into admissible and non-admissible blocks. The non-admissible blocks are stored without compression, while we apply our approximation scheme to the admissible blocks.

2.4 Compressed Representation

For each cluster τ, we denote the interpolation points and Lagrange polynomials corresponding to $\mathcal{I}^{B^\tau}$ by $(x_\nu^\tau)_{\nu \in K}$ and $(\mathcal{L}_\nu^\tau)_{\nu \in K}$ and introduce the matrix $\mathbf{V}^\tau \in \mathbb{R}^{\tau \times K}$ by setting

$$\mathbf{V}_{i\nu}^\tau := \int_\Gamma \mathcal{L}_\nu^\tau(x) \Phi_i(x)\, dx \qquad (i \in \tau,\ \nu \in K). \tag{2.6}$$

The family $(\mathbf{V}^\tau)_{\tau \in T_\mathfrak{I}}$ is called a *cluster basis*. Let $(\tau,\sigma) \in P_{\text{far}}$. We replace the kernel function κ by its interpolant

$$\tilde{\kappa}^{\tau,\sigma} := (\mathcal{I}^{B^\tau} \otimes \mathcal{I}^{B^\sigma})\kappa = \sum_{\nu \in K} \sum_{\mu \in K} \kappa(x_\nu^\tau, x_\mu^\sigma) \mathcal{L}_\nu^\tau(x) \mathcal{L}_\mu^\sigma(y)$$

and get the approximate matrix $\tilde{\mathbf{K}}^{\tau,\sigma} \in \mathbb{R}^{\tau \times \sigma}$ defined by

$$\tilde{\mathbf{K}}_{ij}^{\tau,\sigma} := \sum_{\nu \in K} \sum_{\mu \in K} \kappa(x_\nu^\tau, x_\mu^\sigma) \int_\Gamma \mathcal{L}_\nu^\tau(x) \Phi_i(x)\, dx \int_\Gamma \mathcal{L}_\mu^\sigma(y) \Phi_j(y)\, dy = (\mathbf{V}^\tau \mathbf{S}^{\tau,\sigma} \mathbf{V}^{\sigma\top})_{ij} \tag{2.7}$$

for $i \in \tau$, $j \in \sigma$, where $\mathbf{S}^{\tau,\sigma} \in \mathbb{R}^{K \times K}$ is given by

$$\mathbf{S}_{\nu,\mu}^{\tau,\sigma} := \kappa(x_\nu^\tau, x_\mu^\sigma) \qquad (\nu, \mu \in K). \tag{2.8}$$

The approximation $\tilde{\mathbf{K}} \in \mathbb{R}^{\mathfrak{I} \times \mathfrak{I}}$ of the matrix $\mathbf{K}$ is given by

$$\tilde{\mathbf{K}}_{ij} := \begin{cases} (\mathbf{V}^\tau \mathbf{S}^{\tau,\sigma} \mathbf{V}^{\sigma\top})_{ij} & \text{if } (i,j) \in \tau \times \sigma \text{ for } (\tau,\sigma) \in P_{\text{far}}, \\ \mathbf{K}_{ij} & \text{otherwise,} \end{cases} \qquad (i,j \in \mathfrak{I}).$$

2.5 Fast Matrix-Vector Multiplication

Now that we have derived a compact approximation of the matrix $\mathbf{K}$, we consider the efficient computation of the product of $\tilde{\mathbf{K}}$ and a given vector $u \in \mathbb{R}^\mathfrak{I}$.

The straightforward method is to loop over all blocks $(\tau,\sigma) \in P$ and multiply them by u. This approach is not optimal, since we have to perform the multiplication by $\mathbf{V}^{\sigma\top}$ for each single block of the form $(\tau,\sigma) \in P_{\text{far}}$. In order to remove this redundancy, we split the matrix-vector multiplication into four parts:

1. **Forward transformation:** Compute $\hat{u}_\sigma := \mathbf{V}^{\sigma\top} u|_\sigma$ for all $\sigma \in T_\mathfrak{I}$.
2. **Multiplication:** Compute $\hat{v}_\tau := \sum_{\sigma,(\tau,\sigma) \in P_{\text{far}}} \mathbf{S}^{\tau,\sigma} \hat{u}_\sigma$ for all $\tau \in T_\mathfrak{I}$.
3. **Backward transformation:** Initialize the output vector v by zero and add up the contributions of all clusters: $v|_\tau := v|_\tau + \mathbf{V}^\tau \hat{v}_\tau$.
4. **Nearfield:** Add the uncompressed parts: $v|_\tau := \mathbf{K}|_{\tau \times \sigma} u|_\sigma$ for all $(\tau,\sigma) \in P_{\text{near}}$.

This approach is more efficient than the naive method, but can be improved even further: Let $\tau \in T_\mathfrak{I}$ be a node that is not a leaf. Let $\tau' \in \mathit{sons}(\tau)$. Since our interpolation operators are projections onto $\mathcal{Q}^{d,m}$, we find that

$$\mathcal{L}_\nu^\tau = \mathcal{I}^{B^{\tau'}} \mathcal{L}_\nu^\tau = \sum_{\nu' \in K} \mathcal{L}_\nu^\tau(x_{\nu'}^{\tau'}) \mathcal{L}_{\nu'}^{\tau'} \qquad \text{holds for all } \nu \in K.$$

For a given index $i \in \tau'$, this implies

$$\mathbf{V}_{i\nu}^\tau = \int_\Gamma \mathcal{L}_\nu^\tau(x) \Phi_i(x)\, dx = \sum_{\nu' \in K} \mathcal{L}_\nu^\tau(x_{\nu'}^{\tau'}) \int_\Gamma \mathcal{L}_{\nu'}^{\tau'}(x) \Phi_i(x)\, dx = (\mathbf{V}^{\tau'} \mathbf{B}^{\tau',\tau})_{i\nu}, \tag{2.9}$$

where $\mathbf{B}^{\tau',\tau} \in \mathbb{R}^{K \times K}$ is defined by

$$\mathbf{B}_{\nu'\nu}^{\tau',\tau} := \mathcal{L}_\nu^\tau(x_{\nu'}^{\tau'}) \qquad (\nu, \nu' \in K). \tag{2.10}$$

The equation (2.9) describes an essential property of the basis: The restriction of $\mathbf{V}^\tau$ to the subset τ' belongs to the range of $\mathbf{V}^{\tau'}$, the cluster bases are *nested*.

Using this property, we can derive recursive procedures for performing the first and third step of the matrix-vector multiplication:

```
procedure $fastforward(\sigma,\ u,$ var $(\hat{u}_\sigma)_{\sigma\in T_{\mathcal{I}}})$;
begin if $sons(\sigma)=\emptyset$ then $\hat{u}_\sigma := \mathbf{V}^{\sigma\top} u|_\sigma$
      else for all $\sigma' \in sons(\sigma)$ do
            begin $fastforward(\sigma',\ u,\ \hat{u})$;
                  $\hat{u}_\sigma := \hat{u}_\sigma + \mathbf{B}^{\sigma',\sigma\top}\hat{u}_{\sigma'}$
end         end;
```

```
procedure $fastbackward(\tau,$ var $v,\ (\hat{v}_\tau)_{\tau\in T_{\mathcal{I}}})$;
begin if $sons(\tau)=\emptyset$ then $v|_\tau := \mathbf{V}^\tau \hat{v}_\tau$
      else for all $\tau' \in sons(\tau)$ do
            begin $\hat{v}_{\tau'} := \hat{v}_{\tau'} + \mathbf{B}^{\tau',\tau}_{\tau'}\hat{v}_\tau$;
                  $fastbackward(\tau',\ v,\ \hat{v})$
end         end;
```

In order to stress the similarities of both procedures, we have not included the necessary initialization: Before calling the fast forward transformation, the output coefficients $(\hat{u}_\sigma)_{\sigma\in T_{\mathfrak{J}}}$ have to be set to zero.

The advantage of the recursive procedures is that we have to store the matrices $\mathbf{V}^\tau$ only for the leaves of the cluster tree. For all other clusters, it is sufficient to store the small transfer matrices $\mathbf{B}^{\tau',\tau}$. This leads to a significant reduction in the storage complexity.

Remark 2.2 (Complexity) *In typical situations, building the matrices $\mathbf{B}^{\tau',\tau}$, $\mathbf{V}^\tau$, and $\mathbf{S}^{\tau,\sigma}$ can be done in $\mathcal{O}(Nm^d)$ operations. The matrix-vector multiplication based on the recursive procedures requires $\mathcal{O}(Nm^d)$ operations, too (cf. [3]).*

3 Orthonormalization

Our approximation is created by a d-dimensional interpolation operator, but enters the computations only in the form of boundary integrals over a $(d-1)$-dimensional surface. Therefore we can expect that the used expansion system is too rich and that it should be possible to construct reduced expansion systems, e.g., harmonic polynomials if $\kappa(\cdot,\cdot)$ is the kernel function corresponding to the Laplace equation. We do this by applying algebraic algorithms to the original $\mathcal{H}^2$-matrix.

The algebraic equivalent of the continuous expansion system is the cluster basis $(\mathbf{V}^\tau)_{\tau\in T_{\mathfrak{J}}}$ (cf. (2.6)), which can be considered to represent the Galerkin discretizations of the expansion functions.

Therefore, our goal is to find a reduced cluster basis $(\tilde{\mathbf{V}}^\tau)_{\tau\in T_{\mathfrak{J}}}$ and then to find an approximation of the original $\mathcal{H}^2$-matrix in terms of the new basis.

3.1 Orthonormalized Cluster Basis

Let us fix a cluster $\tau \in T_{\mathfrak{J}}$, and let $\nu \in K$. If the basis function $\mathcal{L}^\tau_\nu$ is redundant, its Galerkin projection can be represented in terms of the projections of the remaining basis functions, therefore we can represent the ν-th column of $\mathbf{V}^\tau$ in terms of the other columns, and this implies that $\mathbf{V}^\tau$ is rank-deficient.

This suggests a simple method for the elimination of redundant expansion functions: We try to orthonormalize $\mathbf{V}^\tau$, i.e., to find a rank $\tilde{k}^\tau \in \mathbb{N}$ and matrix $\mathbf{Q}^\tau \in \mathbb{R}^{K\times\tilde{k}^\tau}$ such that $\tilde{\mathbf{V}}^\tau := \mathbf{V}^\tau\mathbf{Q}^\tau \in \mathbb{R}^{\tau\times\tilde{k}^\tau}$ is orthogonal. The hope is that $\tilde{k}^\tau < \#K$ leads to lower cost without losing accuracy.

The orthogonality of $\tilde{\mathbf{V}}^\tau$ is equivalent to

$$\mathbf{I} = (\tilde{\mathbf{V}}^\tau)^\top\tilde{\mathbf{V}}^\tau = \mathbf{Q}^{\tau\top}(\mathbf{V}^{\tau\top}\mathbf{V}^\tau)\mathbf{Q}^\tau. \tag{3.1}$$

A matrix $\mathbf{Q}^\tau$ satisfying this condition can be found by different algorithms. We use the Schur decomposition

$$\mathbf{P}^\top\mathbf{G}\mathbf{P} = \mathbf{D}$$

of the positive semidefinite symmetric matrix $\mathbf{G} := \mathbf{V}^{\tau\top}\mathbf{V}^\tau$, where $\mathbf{P} \in \mathbb{R}^{K\times k}$ is a square orthogonal matrix (recall that K is the index set corresponding to the original expansion system and k is its cardinality), $\mathbf{D} \in \mathbb{R}^{k\times k}$ is diagonal and the entries of $\mathbf{D}$ are ordered in a monotonously increasing sequence.

If all the diagonal entries of D were positive, we could set $\mathbf{Q}^\tau := \mathbf{P}\mathbf{D}^{-1/2}$ and find

$$\mathbf{Q}^{\tau\top}(\mathbf{V}^{\tau\top}\mathbf{V}^\tau)\mathbf{Q}^\tau = \mathbf{D}^{-1/2}\mathbf{P}^\top\mathbf{G}\mathbf{P}\mathbf{D}^{-1/2} = \mathbf{D}^{-1/2}\mathbf{D}\mathbf{D}^{-1/2} = \mathbf{I},$$

so we would have found a solution for equation (3.1).

For rank-deficient matrices $\mathbf{V}^\tau$, $\mathbf{D}$ will have zero entries, so we have to modify our approach. The idea is to choose those entries of $\mathbf{D}$ that are larger than a given threshold $\epsilon \in \mathbb{R}_{>0}$: We set

$$\tilde{k}^\tau := \min\{\ell \in \{1,\dots,k\} \ :\ \sum_{p=\ell+1}^{k}\mathbf{D}_{pp} \le \epsilon\}$$

and use $\mathbf{D}^\dagger \in \mathbb{R}^{k\times \tilde{k}^\tau}$ given by

$$\mathbf{D}^\dagger_{ij} = \begin{cases} \mathbf{D}_{ij}^{-1/2} & \text{if } i = j, \\ 0 & \text{otherwise} \end{cases} \qquad (i \in \{1,\dots,k\},\ j \in \{1,\dots,\tilde{k}^\tau\})$$

in order to define

$$\tilde{\mathbf{V}}^\tau := \mathbf{V}^\tau \mathbf{P}\mathbf{D}^\dagger.$$

This implies

$$\tilde{\mathbf{V}}^\tau \tilde{\mathbf{V}}^{\tau\top} = \mathbf{V}^\tau \mathbf{P}(\mathbf{D}^\dagger)^2 \mathbf{P}^\top \mathbf{V}^{\tau\top}$$

and therefore

$$\mathbf{V}^{\tau\top}\tilde{\mathbf{V}}^\tau \tilde{\mathbf{V}}^{\tau\top}\mathbf{V}^\tau = \mathbf{G}\mathbf{P}(\mathbf{D}^\dagger)^2\mathbf{P}^\top\mathbf{G} = \mathbf{P}\mathbf{D}(\mathbf{D}^\dagger)^2\mathbf{D}\mathbf{P}^\top.$$

Due to orthogonality, the best approximation of $\mathbf{V}^\tau$ in the range of $\tilde{\mathbf{V}}^\tau$ is given by $\tilde{\mathbf{V}}^\tau\tilde{\mathbf{V}}^{\tau\top}\mathbf{V}^\tau$ and satisfies the error estimate

$$\begin{aligned} \|\mathbf{V}^\tau - \tilde{\mathbf{V}}^\tau\tilde{\mathbf{V}}^{\tau\top}\mathbf{V}^\tau\|_F^2 &= \operatorname{tr}(\mathbf{V}^{\tau\top}\mathbf{V}^\tau - \mathbf{V}^{\tau\top}\tilde{\mathbf{V}}^\tau\tilde{\mathbf{V}}^{\tau\top}\mathbf{V}^\tau) \\ &= \operatorname{tr}(\mathbf{P}(\mathbf{D} - \mathbf{D}(\mathbf{D}^\dagger)^2\mathbf{D})\mathbf{P}^\top) = \operatorname{tr}(\mathbf{D} - \mathbf{D}(\mathbf{D}^\dagger)^2\mathbf{D}) \le \epsilon \end{aligned}$$

($\|\cdot\|_F$ is the Frobenius norm).

3.2 Nested Basis

Applying the straightforward orthonormalization described above to all clusters $\tau \in T_{\mathfrak{I}}$ will give us a reduced basis, but this basis will no longer be nested, i.e., the equation (2.9) will no longer hold, so we would not be able to use the fast matrix-vector multiplication algorithm.

Therefore we have to introduce a further modification: Let $\tau \in T_{\mathfrak{I}}$ be a cluster with $sons(\tau) \neq \emptyset$. Due to (2.9), we have

$$\mathbf{V}^\tau|_{\tau'\times K} = \mathbf{V}^{\tau'}\mathbf{B}^{\tau',\tau} \qquad \text{for all } \tau' \in sons(\tau). \tag{3.2}$$

We want to find a reduced matrix $\tilde{\mathbf{V}}^\tau$ satisfying a similar equation for reduced transfer matrices $\tilde{\mathbf{B}}^{\tau',\tau}$.

Suppose we have already computed $\tilde{\mathbf{V}}^{\tau'}$ for all $\tau' \in sons(\tau)$ of τ. We approximate $\mathbf{V}^{\tau'}$ in (3.2) in terms of $\tilde{\mathbf{V}}^{\tau'}$, i.e., we apply the orthogonal projection to the range of $\tilde{\mathbf{V}}^{\tau'}$ to both sides of the equation:

$$\tilde{\mathbf{V}}^{\tau'}\tilde{\mathbf{V}}^{\tau'\top}\mathbf{V}^\tau|_{\tau'\times K} = \tilde{\mathbf{V}}^{\tau'}\tilde{\mathbf{V}}^{\tau'\top}\mathbf{V}^{\tau'}\mathbf{B}^{\tau',\tau}.$$

Let $\mathbf{W}^{\tau'} := (\tilde{\mathbf{V}}^{\tau'})^\top\mathbf{V}^{\tau'}$, and let $\widehat{\mathbf{V}}^\tau \in \mathbb{R}^{\tau\times K}$ be defined by

$$\widehat{\mathbf{V}}^\tau|_{\tau'\times K} := \tilde{\mathbf{V}}^{\tau'}\mathbf{W}^{\tau'}\mathbf{B}^{\tau',\tau} \tag{3.3}$$

for all $\tau' \in sons(\tau)$. $\widehat{\mathbf{V}}^\tau$ is well-defined since the sons $\tau' \in sons(\tau)$ are disjoint and their union is τ.

The matrix $\widehat{\mathbf{V}}^\tau$ is the orthogonal projection of $\mathbf{V}^\tau$ onto the space spanned by the ranges of the matrices $\tilde{\mathbf{V}}^{\tau'}$ corresponding to the sons of τ, i.e., $\widehat{\mathbf{V}}^\tau$ is the best approximation of $\mathbf{V}^\tau$ we can get without giving up nestedness.

Now we can apply the same orthogonalization procedure as before to $\widehat{\mathbf{V}}^\tau$ instead of $\mathbf{V}^\tau$ in order to find a matrix $\mathbf{Q}^\tau$ satisfying

$$\mathbf{I} = (\tilde{\mathbf{V}}^\tau)^\top\tilde{\mathbf{V}}^\tau = \mathbf{Q}^{\tau\top}(\widehat{\mathbf{V}}^{\tau\top}\widehat{\mathbf{V}}^\tau)\mathbf{Q}^\tau.$$

The matrix $\tilde{\mathbf{V}}^\tau$ is given by

$$\tilde{\mathbf{V}}^\tau := \widehat{\mathbf{V}}^\tau\mathbf{Q}^\tau,$$

and this implies

$$\tilde{\mathbf{V}}^\tau|_{\tau'\times\tilde{k}^\tau} = \widehat{\mathbf{V}}^\tau|_{\tau'\times K}\mathbf{Q}^\tau = \tilde{\mathbf{V}}^{\tau'}\mathbf{W}^{\tau'}\mathbf{B}^{\tau',\tau}\mathbf{Q}^\tau = \tilde{\mathbf{V}}^{\tau'}\tilde{\mathbf{B}}^{\tau',\tau}$$

for $\tilde{\mathbf{B}}^{\tau',\tau} := \mathbf{W}^{\tau'}\mathbf{B}^{\tau',\tau}\mathbf{Q}^\tau$, i.e., the new cluster basis $(\tilde{\mathbf{V}}^\tau)_{\tau\in T_{\mathfrak{I}}}$ is nested (cf. (2.9)).

The Gram matrix $\widehat{\mathbf{V}}^{\tau\top}\widehat{\mathbf{V}}^\tau$ used in the computation of $\tilde{\mathbf{B}}^{\tau',\tau}$ can be constructed by means of the equation

$$\widehat{\mathbf{V}}^{\tau\top}\widehat{\mathbf{V}}^\tau = \sum_{\tau'\in sons(\tau)} \mathbf{B}^{\tau',\tau\top}\mathbf{W}^{\tau'\top}\tilde{\mathbf{V}}^{\tau'\top}\tilde{\mathbf{V}}^{\tau'}\mathbf{W}^{\tau'}\mathbf{B}^{\tau',\tau} = \sum_{\tau'\in sons(\tau)} \mathbf{B}^{\tau',\tau\top}\mathbf{W}^{\tau'\top}\mathbf{W}^{\tau'}\mathbf{B}^{\tau',\tau}$$

due to the orthogonality of $\tilde{\mathbf{V}}^{\tau'}$.

By splitting the matrix $\mathbf{W}^\tau$ and using the nestedness of the bases $(\mathbf{V}^\tau)_{\tau\in T_{\mathfrak{I}}}$ and $(\tilde{\mathbf{V}}^\tau)_{\tau\in T_{\mathfrak{I}}}$, we find that the matrix $\mathbf{W}^\tau$ can be represented in the form

$$\mathbf{W}^\tau = \tilde{\mathbf{V}}^{\tau\top}\mathbf{V}^\tau = \sum_{\tau'\in sons(\tau)} \tilde{\mathbf{B}}^{\tau',\tau\top}\tilde{\mathbf{V}}^{\tau'\top}\mathbf{V}^{\tau'}\mathbf{B}^{\tau',\tau} = \sum_{\tau'\in sons(\tau)} \tilde{\mathbf{B}}^{\tau',\tau\top}\mathbf{W}^{\tau'}\mathbf{B}^{\tau',\tau}; \tag{3.4}$$

hence, this computation requires only the matrices $\mathbf{W}^{\tau'}$ corresponding to the sons τ' of τ and the transfer matrices $\mathbf{B}^{\tau',\tau}$ and $\tilde{\mathbf{B}}^{\tau',\tau}$.

The following algorithm computes the matrices $\tilde{\mathbf{V}}^\tau$ for leaves $\tau\in T_{\mathfrak{I}}$ and the transfer matrices $\tilde{\mathbf{B}}^{\tau',\tau}$ for the remaining clusters $\tau\in T_{\mathfrak{I}}$ with $\tau'\in sons(\tau)$:

```
procedure $orthonormalize(\tau)$;
begin if $sons(\tau)=\emptyset$ then                                   {$treatment\ of\ leaves$}
      begin $\mathbf{G} := \mathbf{V}^{\tau\top}\mathbf{V}^\tau$;                                   {$build\ Gram\ matrix$}
            Find $\mathbf{Q}^\tau$ with $\mathbf{Q}^{\tau\top}\mathbf{G}\mathbf{Q}^\tau = \mathbf{I}$;
            $\tilde{\mathbf{V}}^\tau := \mathbf{V}^\tau\mathbf{Q}^\tau$;                                         {$new\ basis\ for\ \tau$}
            $\mathbf{W}^\tau := \tilde{\mathbf{V}}^{\tau\top}\mathbf{V}^\tau$                            {$update\ transformation\ matrix$}
      end else
      begin for all $\tau'\in sons(\tau)$ do $orthonormalize(\tau')$;                        {$recursion$}
            $\mathbf{G} := 0$;                                                 {$build\ projected\ Gram\ matrix$}
            for all $\tau'\in sons(\tau)$ do $\mathbf{G} := \mathbf{G} + \mathbf{B}^{\tau',\tau\top}\mathbf{W}^{\tau'\top}\mathbf{W}^{\tau'}\mathbf{B}^{\tau',\tau}$;
            Find $\mathbf{Q}^\tau$ with $\mathbf{Q}^{\tau\top}\mathbf{G}\mathbf{Q}^\tau = \mathbf{I}$;
            for all $\tau'\in sons(\tau)$ do $\tilde{\mathbf{B}}^{\tau',\tau} := \mathbf{W}^{\tau'}\mathbf{B}^{\tau',\tau}\mathbf{Q}^\tau$;               {$new\ basis\ for\ \tau$}
            $\mathbf{W}^\tau := 0$;                                          {$update\ transformation\ matrix$}
            for all $\tau'\in sons(\tau)$ do $\mathbf{W}^\tau := \mathbf{W}^\tau + \tilde{\mathbf{B}}^{\tau',\tau\top}\mathbf{W}^{\tau'}\mathbf{B}^{\tau',\tau}$
end   end;
```

This procedure is local, i.e., only the matrix $\mathbf{V}^\tau$ is needed for the computation if τ is a leaf, and only the matrices $\mathbf{B}^{\tau',\tau}$ for $\tau'\in sons(\tau)$ are needed if τ is not a leaf. By using temporary variables that are initialized at the beginning of the procedure (cf. (2.6) and (2.10)), we do not need to store the entire original cluster basis.

3.3 Conversion of the Coefficient Matrices

Let $(\tau,\sigma)\in P_{\text{far}}$. The submatrix corresponding to this block of an $\mathcal{H}^2$-matrix is given in the form $\mathbf{V}^\tau\mathbf{S}^{\tau,\sigma}\mathbf{V}^{\sigma\top}$ (cf. (2.7)).

In order to find a representation of this block with respect to the new cluster basis $(\tilde{\mathbf{V}}^\tau)_{\tau\in T_{\mathfrak{I}}}$, we once more use the orthogonal projection and get

$$\tilde{\mathbf{V}}^\tau\tilde{\mathbf{V}}^{\tau\top}\mathbf{V}^\tau\mathbf{S}^{\tau,\sigma}\mathbf{V}^{\sigma\top}\tilde{\mathbf{V}}^\sigma\tilde{\mathbf{V}}^{\sigma\top} = \tilde{\mathbf{V}}^\tau\tilde{\mathbf{S}}^{\tau,\sigma}\tilde{\mathbf{V}}^{\sigma\top}$$

as the best approximation with

$$\tilde{\mathbf{S}}^{\tau,\sigma} := \mathbf{V}^{\tau\top}\mathbf{V}^\tau\mathbf{S}^{\tau,\sigma}\mathbf{V}^{\sigma\top}\tilde{\mathbf{V}}^\sigma = \mathbf{W}^\tau\mathbf{S}^{\tau,\sigma}\mathbf{W}^{\sigma\top},$$

so we need only the matrices $(\mathbf{W}^\tau)_{\tau\in T_{\mathfrak{I}}}$ computed as a by-product in the basis orthonormalization algorithm.

3.4 Complete Algebraic Recompression

The orthogonalization algorithm considers only the expansion system itself, but not the kernel function we intend to approximate. Therefore we can expect improved results if we include the coefficient matrices $\mathbf{S}^{\tau,\sigma}$ in addition to the cluster basis.

This is done by the algorithm introduced in [2] for dense and hierarchical matrices. Since $\mathcal{H}^2$-matrices are a specialization of hierarchical matrices, we could convert the $\mathcal{H}^2$-matrix representation into the form of an $\mathcal{H}$-matrix representation and apply the algorithm from [2] directly. For the conversion, we would have to compute the matrices $(\mathbf{V}^\tau)_{\tau\in T_\mathcal{J}}$ for *all* clusters $\tau \in T_\mathcal{J}$, not only for the leaves, i.e., we could not benefit from the more compact recursive representation based on equation (2.9) .

Instead of computing the matrices $(\mathbf{V}^\tau)_{\tau\in T_\mathcal{J}}$ in advance and using them to expand the factorized form $\mathbf{V}^\tau\mathbf{S}^{\tau,\sigma}\mathbf{V}^{\sigma\top}$ of the $\mathcal{H}^2$-matrix blocks to the form used in the $\mathcal{H}$-matrix conversion algorithm, we keep the blocks factorized as long as possible and expand them during the course of the recursion: We use the set of blocks related to ancestors of τ given by

$$A^\tau := \{\sigma \in T_\mathcal{J} \ : \ \exists\tau_0 \in T_\mathcal{J} : \tau \subseteq \tau_0, (\tau_0,\sigma) \in P_{\text{far}}\}$$

and store the intermediate results of the transformation of these blocks in a family $(\mathbf{C}^{\tau,\sigma})_{\sigma\in A^\tau}$ of auxiliary matrices. After a suitable basis for a cluster has been found, all the blocks are transformed into the new basis and the results are stored in another family $(\widehat{\mathbf{C}}^{\tau,\sigma})_{\sigma\in A^\tau}$ of auxiliary matrices. This leads to the following algorithm:

```
procedure $recompression(\tau,\ A^\tau)$;
begin if $sons(\tau)=\emptyset$ then for all $\sigma\in A^\tau$ do $\hat{\mathbf{C}}^{\tau,\sigma} := \mathbf{C}^{\tau,\sigma}$      {no conversion for leaves}
      else
      begin let $sons(\tau)=\{\tau_1,\dots\tau_s\}$; for $i:=1$ to $s$ do      {compute ancestor blocks}
            begin $R^{\tau_i} := \{\sigma\in T_\mathcal{I} : (\tau_i,\sigma)\in P_{\text{far}}\}$; $A^{\tau_i} := A^\tau \cup R^{\tau_i}$;
                  for all $\sigma\in A^\tau$ do $\mathbf{C}^{\tau_i,\sigma} := \mathbf{B}^{\tau_i,\tau}\mathbf{C}^{\tau,\sigma}$;
                  for all $\sigma\in R^{\tau_i}$ do $\mathbf{C}^{\tau_i,\sigma} := \mathbf{S}^{\tau_i,\sigma}$;
                  $recompression(\tau_i,\ A^{\tau_i})$      {determine cluster basis for sons}
            end;
            for $i:=1$ to $s$ do for $j:=1$ to $s$ do      {compute Gram matrix for all sons}
            begin $\mathbf{G}_{ij} := 0$;
                  for all $\sigma\in A^\tau$ do $\mathbf{G}_{ij} := \mathbf{G}_{ij} + \hat{\mathbf{C}}^{\tau_i,\sigma\top}\mathbf{T}^\sigma\hat{\mathbf{C}}^{\tau_j,\sigma}$
            end;
            find an orthogonal $\tilde{k}^\tau$-column matrix $\mathbf{Q} = \left(\mathbf{Q}_1^\top,\dots,\mathbf{Q}_s^\top\right)^\top$ maximizing
                  $\left\| \left(\mathbf{Q}_1^\top,\dots,\mathbf{Q}_s^\top\right) \begin{pmatrix} \mathbf{G}_{11} & \dots & \mathbf{G}_{1s} \\ \vdots & \ddots & \vdots \\ \mathbf{G}_{s1} & \dots & \mathbf{G}_{ss} \end{pmatrix} \begin{pmatrix} \mathbf{Q}_1 \\ \vdots \\ \mathbf{Q}_s \end{pmatrix} \right\|_F$;
            for all $\sigma\in A^\tau$ do $\hat{\mathbf{C}}^{\tau,\sigma} := 0$;      {convert blocks to the new basis}
            for $i:=1$ to $s$ do
            begin $\tilde{\mathbf{B}}^{\tau_i,\tau} := \mathbf{Q}_i$;
                  for all $\sigma\in A^\tau$ do $\hat{\mathbf{C}}^{\tau,\sigma} := \hat{\mathbf{C}}^{\tau,\sigma} + (\tilde{\mathbf{B}}^{\tau_i,\tau})^\top\mathbf{C}^{\tau_i,\sigma}$
end   end   end;
```

The matrices $(\mathbf{T}^\sigma)_{\sigma\in T_\mathcal{J}}$ appearing in this procedure are defined by $\mathbf{T}^\sigma := \mathbf{V}^{\sigma\top}\mathbf{V}^\sigma$ and can be computed by a recursion similar to (3.4) .

4 Numerical Experiments

For our experiments, we consider the kernel function

$$\kappa(x,y) := \frac{1}{4\pi\|x-y\|}$$

corresponding to the three-dimensional single layer potential operator. The domain Ω is the unit ball in $\mathbb{R}^3$ and therefore Γ is the unit sphere in three dimensions.

We approximate Γ by a regular triangulation consisting of plane triangles and use piecewise constant basis functions for our Galerkin discretization.

The original discretization is performed for an interpolation order of 4, and we choose $\epsilon = 10^{-4}$ as the threshold for the algebraic compression algorithm.

We will first consider the speed of our algorithms. The following table lists the time needed[1] for building the different $\mathcal{H}^2$-matrix approximations and for performing the matrix-vector multiplications:

N	original		orthogonalized		recompressed	
	build/s	MVM/s	build/s	MVM/s	build/s	MVM/s
512	4.33	0.07	5.42	0.01	6.17	0.01
2048	20.16	0.48	25.83	0.09	30.46	0.04
8192	83.50	1.71	108.24	0.45	223.51	0.24
32768	333.25	6.87	435.54	1.78	1163.62	1.02
131072	1315.24	27.05	1718.66	7.42	5683.39	4.08
524288					27366.03	16.07

Table 1: Time in seconds for setup ('build') and matrix-vector multiplication ('MVM')

We can see that both compression techniques lead to a significant reduction in the time for the matrix-vector multiplication: For the simple orthogonalization algorithm the matrix-vector multiplication is speeded up by a factor of almost 4, while the full recompression even gives us a factor of more than 6.

On the other hand, building the fully recompressed $\mathcal{H}^2$-matrix requires much more time than building the $\mathcal{H}^2$-matrix with orthogonalized cluster basis. This is not surprising, since the full recompression algorithm has to consider *all* admissible blocks of the $\mathcal{H}^2$-matrix, while the orthogonalization algorithm works only based on the cluster basis.

Next, we consider the amount of memory required for storing the original and compressed $\mathcal{H}^2$-approximations:

N	original	orthogonalized	recompressed
512	92108	6633	3527
2048	139020	15057	5015
8192	152422	19873	5848
32768	152618	21235	6171
131072	146278	20345	6138

Table 2: Memory requirement in bytes per degree of freedom

The last three columns give the number of bytes of storage needed per degree of freedom. In this table, the advantage of the full recompression method is obvious: It reduces the memory requirements by more than 95% compared to the original method, while the orthogonalization algorithm reaches only 86%.

Of course, we are not only interested in fast algorithms, we also need the results to be sufficiently precise. We approximate the operator norm of matrices by starting with a random vector and performing 100 steps of the power iteration. The values of the relative error $\|\mathbf{K} - \tilde{\mathbf{K}}\|_2/\|\mathbf{K}\|_2$ are collected in the following table:

N	original	orthogonalized	recompressed
512	4.53033_{-5}	4.52851_{-5}	4.53183_{-5}
2048	5.08654_{-5}	5.11205_{-5}	5.10336_{-5}
8192	6.63957_{-5}	6.65225_{-5}	6.64592_{-5}
32768	7.22294_{-5}	7.22293_{-5}	7.23569_{-5}

Table 3: Relative approximation error

Here, the columns "original", "orthogonalized" and "recompressed" give the norm of the difference between the densely populated matrix and the original $\mathcal{H}^2$-matrix, the $\mathcal{H}^2$-matrix with orthogonalized cluster bases and the recompressed $\mathcal{H}^2$-matrix.

Obviously, the additional errors introduced by orthogonalization and recompression are negligible. This means that both algorithms yield a significant reduction in the computational complexity without affecting the precision. The compression ratio of the full recompression procedure is much better than that of the simple orthogonalization method, but this advantage is complemented by the significantly higher complexity.

[1] All computations were performed on Sun Ultra 3cu processors running at 900 MHz.

References

[1] S. Börm, L. Grasedyck, and W. Hackbusch, *Introduction to hierarchical matrices with applications*, Engineering Analysis with Boundary Elements, 27 (2003), pp. 405–422.

[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] 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. Accepted by Numerische Mathematik.

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

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

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

[8] 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.

[9] 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.

[10] 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.

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

Steffen Börm
Max-Planck-Institut für Mathematik in den Naturwissenschaften
Inselstraße 22–26, D-04103 Leipzig
eMail: sbo@mis.mpg.de

Wolfgang Hackbusch
Max-Planck-Institut für Mathematik in den Naturwissenschaften
Inselstraße 22–26, D-04103 Leipzig
eMail: wh@mis.mpg.de