Max-Planck-Institut
für Mathematik
in den Naturwissenschaften
Leipzig

Hybrid Cross Approximation of Integral Operators

(revised version: March 2005)

by

Steffen Börm and Lars Grasedyck

Preprint no.: 68 2004

Hybrid Cross Approximation of Integral Operators

Steffen Börm Lars Grasedyck

March 17, 2005

The efficient treatment of dense matrices arising, e.g., from the finite element discretisation of integral operators requires special compression techniques. In this article we use the $\mathcal{H}$-matrix representation that approximates the dense stiffness matrix in admissible blocks (corresponding to subdomains where the underlying kernel function is smooth) by low-rank matrices. The low-rank matrices are assembled by a new hybrid algorithm (HCA) that has the same proven convergence as standard interpolation but also the same efficiency as the (heuristic) adaptive cross approximation (ACA).

Keywords: hierarchical matrices – data-sparse approximations – formatted matrix operations – fast solvers – preconditioning – boundary elements

1 Introduction

The efficient treatment of dense matrices arising, e.g., from the finite element discretisation of integral operators requires special compression techniques to avoid the quadratic cost for the assembly and storage.

The standard techniques in this field of research include but are not limited to panel clustering [19, 22], multipole expansions [21, 16], interpolation [5] and (adaptive) cross approximation (ACA) [23, 1, 2]. If the underlying geometry can be described by a small number of smooth maps, wavelet techniques can be used in order to compress the resulting dense matrix [9].

Panel clustering often uses an explicit Taylor series expansion of the kernel function, which implies that suitable recursion formulae have to be derived analytically for any given kernel function. This disadvantage can be overcome by the use of general interpolation formulae. However, the separation rank produced by these methods is rather large since one neglects the special structure of the kernel function. Multipole expansion on the other hand exploits this special structure but requires an explicit expansion of the kernel. Thus, this method is limited to the standard kernels where these are known.

The (adaptive) cross approximation is an algebraic method where no expansion of the kernel is needed. However, a convergence proof does only exist for Nystrøm discretisations of the single layer potential. Moreover, a simple counterexample shows that this method may fail for more general kernel functions and geometries with edges.

Our contribution is a new hybrid method that combines the ACA algorithm with the interpolation based separation of the kernel function. For the new method we are able to rigorously prove convergence, both for single layer and double layer potentials of asymptotically smooth kernels as well as Nystrøm, collocation or Galerkin boundary element formulations.

A convenient format to store the arising matrices is the $\mathcal{H}$-matrix format [17, 18, 13, 15] which is at the same time useful to construct an efficient preconditioner or even an accurate inverse.

The rest of this article is organised as follows: in Section 2 we introduce a simple model problem, describe in short the $\mathcal{H}$-matrix format and summarise two standard compression methods. In Section 3 we introduce the new hybrid cross approximation (HCA) algorithm, estimate the approximation error in Section 4 and provide numerical examples that underline the theoretical results in the last Section 5.

2 The $\mathcal{H}$-Matrix Format

2.1 Model Problem: Integral Equation

We consider a Fredholm integral operator of the form

$$\mathcal{G}[u](x) = \int_\Omega g(x,y)u(y)\,\mathrm{d}y \tag{2.1}$$

on a submanifold or subdomain Ω of $\mathbb{R}^3$ with a kernel function

$$g : \Omega \times \Omega \to \mathbb{R}.$$

The kernel function might be (but is not limited to) the classical single or double layer kernel for the Laplacian on a manifold Ω:

$$g^{\mathrm{SLP}}(x,y) := \frac{1}{4\pi\|x-y\|},$$
$$g^{\mathrm{DLP}}(x,y) := \frac{\langle x-y, n(y)\rangle}{4\pi\|x-y\|^3} = \frac{\partial g^{\mathrm{SLP}}}{\partial n(y)}(x,y).$$

$\mathcal{H}$-matrices are based on the fact that the typical kernel functions can be approximated by local degenerate expansions. We assume that the kernel function g results from applying a partial differential operator to a sufficiently smooth generator function

$$\gamma : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R},$$

i.e., that there are coefficients

$$c^x, c^y : \Omega \to \mathbb{R}^3, \qquad c_0^x, c_0^y : \Omega \to \mathbb{R} \tag{2.2}$$

such that

$$g = D_x D_y \gamma \tag{2.3}$$

holds for differential operators

$$D_x := \langle c^x, \nabla_x \rangle + c_0^x = \sum_{i=1}^{3} c_i^x \partial_{x_i} + c_0^x,$$

$$D_y := \langle c^y, \nabla_y \rangle + c_0^y = \sum_{j=1}^{3} c_j^y \partial_{y_j} + c_0^y.$$

In typical applications, the generator function γ is *asymptotically smooth*, i.e., there exist constants C_{as1} and C_{as2} and a singularity degree $\sigma \geq 0$ such that for all $\alpha, \beta \in \mathbb{N}_0^d$ the inequality

$$|\partial_x^\alpha \partial_y^\beta \gamma(x,y)| \leq C_{\mathrm{as1}} (C_{\mathrm{as2}} \|x-y\|)^{-|\alpha+\beta|-\sigma} (\alpha+\beta)! \tag{2.4}$$

holds. In the case of the classical Laplace operator, the function $\gamma = g^{\mathrm{SLP}}$ satisfies this property with $C_{\mathrm{as1}} = C_{\mathrm{as2}} = 1$, and setting $c^x(x) = 0$, $c^y(y) = n(y)$, $c_0^x(x) = 1$ and $c_0^y(x) = 0$ yields $g^{\mathrm{DLP}} = D_x D_y \gamma$.

A standard Galerkin discretisation of $\mathcal{G}$ for a basis $(\varphi_i)_{i \in I}$, $I = \{1, \ldots, n\}$, requires the computation of the stiffness matrix $G \in \mathbb{R}^{n \times n}$ given by

$$G_{ij} := \int_\Omega \int_\Omega \varphi_i(x) g(x,y) \varphi_j(y) \, \mathrm{d}y \, \mathrm{d}x. \tag{2.5}$$

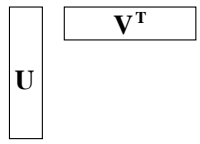
Since the support of the kernel g is in general not local, one expects a dense matrix G. The algorithmic complexity for computing and storing a dense matrix is quadratic in the number of degrees of freedom, so for large problem dimensions data-sparse representations or approximations have to be used.

2.2 Low-Rank Approximation

Hierarchical matrices [17, 18, 6, 15, 5] are the algebraic counterpart of the local degenerate approximations used in panel clustering and multipole techniques. On the discrete level, replacing a function by a degenerate expansion means that blocks of the matrix are approximated by low-rank matrices.

Definition 2.1 ($R(k)$-Matrix Format) *Let $X \in \mathbb{R}^{n \times m}$, $k \in \mathbb{N}$ such that the rank of X is bounded by k. An $R(k)$-matrix representation of X is a factorisation of the form*

$$X = AB^T$$



with matrices $A \in \mathbb{R}^{n \times k}, B \in \mathbb{R}^{m \times k}$.

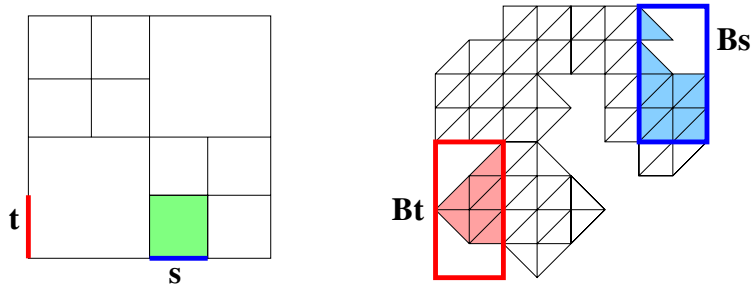


Figure 1: The block $t \times s \subset I \times I$ corresponds to a subset $\Omega_t \times \Omega_s$ of $\Omega \times \Omega$.

The storage requirements for an $R(k)$-matrix are $k(n+m)$ instead of the quadratic cost nm for standard full matrices. Since the rank k is expected to be $k \approx \log(n)$ this is a data-sparse representation of the matrix X. Moreover, the matrix-vector multiplication $w := Xv$ can be split into two multiplications $x := B^T v$, $w := Ax$ so that only $2k(n+m)-k-n$ additions and multiplications of real numbers are necessary to perform the exact matrix-vector multiplication.

2.2.1 Low-Rank Approximation by Interpolation

Let us now investigate how a low-rank approximation of the matrix G (cf. (2.5)) can be constructed. Let $t \times s \subset I \times I$. We set

$$\Omega_t := \cup_{i \in t} \operatorname{supp}(\varphi_i), \qquad \Omega_s := \cup_{j \in s} \operatorname{supp}(\varphi_j)$$

and fix axially parallel boxes B_t and B_s such that $\Omega_t \subseteq B_t$ and $\Omega_s \subseteq B_s$ hold. We require that the *strong η-admissibility condition*

$$\max\{\operatorname{diam}(B_t), \operatorname{diam}(B_s)\} \leq \eta \operatorname{dist}(B_t, B_s) \tag{2.6}$$

holds (cf. Figure 1). A low-rank approximation of the corresponding matrix block $G|_{t \times s}$ can be computed by interpolation: let $\tilde{\gamma}$ be the polynomial constructed by m-th order tensor-product Chebyshev interpolation of γ.

Let $M := (m+1)^3$. Since $\tilde{\gamma}$ has been constructed by tensor-product interpolation, there are interpolation points $(x_\nu^t)_{\nu=1}^M$ in B^t and $(x_\mu^s)_{\mu=1}^M$ in B^s with corresponding Lagrange polynomials $(\mathcal{L}_\nu^t)_{\nu=1}^M$ and $(\mathcal{L}_\mu^s)_{\mu=1}^M$ such that

$$\tilde{\gamma}(x,y) = \sum_{\nu=1}^{M} \sum_{\mu=1}^{M} \gamma(x_\nu^t, x_\mu^s) \mathcal{L}_\nu^t(x) \mathcal{L}_\mu^s(y) \tag{2.7}$$

holds. By applying D_x and D_y to $\tilde{\gamma}$, we can construct an approximation

$$\begin{aligned} \tilde{g}(x,y) &:= D_x D_y \tilde{\gamma}(x,y) \\ &= \sum_{\nu=1}^{M} \sum_{\mu=1}^{M} \gamma(x_\nu^t, x_\mu^s) (D_x \mathcal{L}_\nu^t)(x) (D_y \mathcal{L}_\mu^s)(y) \end{aligned} \tag{2.8}$$

of g, and replacing g by $\tilde{g}$ in (2.5) yields an approximation $G \approx \widetilde{G}$ with entries

$$\begin{aligned}\widetilde{G}_{ij} &:= \int_\Omega \int_\Omega \varphi_i(x)\tilde{g}(x,y)\varphi_j(y)\,\mathrm{d}y\,\mathrm{d}x \\ &= \sum_{\nu=1}^{M}\sum_{\mu=1}^{M} \gamma(x_\nu^t, x_\mu^s) \int_\Omega \varphi_i(x)(D_x\mathcal{L}_\nu^t)(x)\,\mathrm{d}x \int_\Omega \varphi_j(y)(D_y\mathcal{L}_\mu^s)(y)\,\mathrm{d}y \\ &= (U_t S_{t,s}(V_s)^T)_{ij} \end{aligned} \tag{2.9}$$

for $i \in t$, $j \in s$. The matrices $U_t \in \mathbb{R}^{t\times M}$, $V_s \in \mathbb{R}^{s\times M}$ and $S_{t,s} \in \mathbb{R}^{M\times M}$ are given by

$$(U_t)_{i\nu} := \int_\Omega \varphi_i(x)(D_x\mathcal{L}_\nu^t)(x)\,\mathrm{d}x, \quad (V_s)_{j\mu} := \int_\Omega \varphi_j(y)(D_y\mathcal{L}_\mu^s)(y)\,\mathrm{d}y, \tag{2.10}$$

$$(S_{t,s})_{\nu\mu} := \gamma(x_\nu^t, x_\mu^s). \tag{2.11}$$

Since the coupling matrix $S_{t,s}$ is of dimension M, the rank of the factorised matrix $U_t S_{t,s} V_s^T$ is bounded by M, i.e., the matrix block $G|_{t\times s}$ can be approximated by the $R(M)$-matrices $(U_t S_{t,s})V_s^T$ or $U_t(V_s S_{t,s}^T)^T$.

We note that U_t depends only on t and V_s depends only on s, while the coupling matrix $S_{t,s}$ depends on t, s and the generator function γ. Therefore, U_t and V_s can be constructed by standard quadrature algorithms, while $S_{t,s}$ contains pointwise evaluations of the generator function.

2.2.2 Low-Rank Approximation by ACA

We will now introduce the ACA algorithm for a matrix

$$X_{ij} := g(x_i, y_j), \qquad i = 1,\ldots,n, \quad j = 1,\ldots,m.$$

The procedure is given in detail in Algorithm 1.

For matrix entries X_{ij} that stem from the evaluation of an asymptotically smooth function, as it is the case for the matrix $X = S_{t,s}$ above, [1, Theorem 4] states

$$|X_{ij} - (AB^T)_{ij}| = \mathcal{O}\left(2^k(\eta/2)^{\sqrt[3]{k}}\right),$$

where η is the admissibility parameter and k is the rank. In practice, the factor 2^k does not appear and the convergence rate for the single layer kernel seems to be $\mathcal{O}\left((\eta/2)^{\sqrt{k}}\right)$, so we observe that $k = \log(\varepsilon)^2$ is often sufficient for a relative approximation error of $\mathcal{O}(\varepsilon)$.

The ACA algorithm with partial pivoting (as it is used in [2] or modified as in [11]) may fail if the kernel function is not asymptotically smooth with respect to both variables. In order to prove this we give a simple counterexample.

Example 2.2 (Counterexample for Partial Pivoting) *Let $\Omega_t = \Omega_{t_1} \cup \Omega_{t_2}$ and let $\Omega_s = \Omega_{s_1} \cup \Omega_{s_2}$ with*

Algorithm 1 ACA with partial pivoting

procedure ACA(X, **var** A, B)
Choose an initial pivot index i_1^*
$k := 1$
repeat
Compute the entries of the vector $b_k \in \mathbb{R}^m$ by

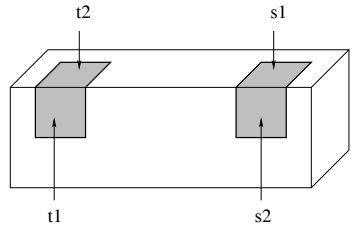
$$(b_k)_j := X_{i_k^*,j} - \sum_{\mu=1}^{k-1} (a_\mu)_{i_k^*} (b_\mu)_j.$$

Determine an index j_k^* that maximises $\delta := |(b_k)_{j_k^*}|$
Compute the entries of the vector $a_k \in \mathbb{R}^n$ by

$$(a_k)_i := \left(X_{i,j_k^*} - \sum_{\mu=1}^{k-1} (a_\mu)_i (b_\mu)_{j_k^*} \right) / (b_k)_{j_k^*}$$

Determine the next pivot index $i_{k+1}^* \neq i_k^*$ that maximises $\delta := |(a_k)_{i_{k+1}^*}|$
$k := k + 1$
until $\|a_k\|_2 \|b_k\|_2 \leq \epsilon \|a_1\|_2 \|b_1\|_2$

$\Omega_{t_1} := [0,1] \times [0,1] \times \{0\}$,
$\Omega_{t_2} := [0,1] \times \{0\} \times [0,1]$,
$\Omega_{s_1} := [4,5] \times \{0\} \times [0,1]$,
$\Omega_{s_2} := [4,5] \times [0,1] \times \{0\}$.



We have $\operatorname{diam}(\Omega_t) = \operatorname{diam}(\Omega_s) = \sqrt{3}$ *and* $\operatorname{dist}(\Omega_t, \Omega_s) = 3$*, so the domains are admissible for all* $\eta > 1/\sqrt{3}$.

Let us fix $n, m \in \mathbb{N}$ *and points* $(x_i)_{i=1}^{n+m}$ *and* $(y_j)_{j=1}^{n+m}$ *satisfying*

$$x_i \in \begin{cases} \Omega_{t_1} & \text{if } i \leq n \\ \Omega_{t_2} & \text{otherwise} \end{cases} \qquad y_j \in \begin{cases} \Omega_{s_1} & \text{if } j \leq n \\ \Omega_{s_2} & \text{otherwise} \end{cases}$$

for $i, j \in \{1, \ldots, n+m\}$. *Evaluating the double layer potential* g^{DLP} *(which is asymptotically smooth in the* x *variable, but not in the* y *variable) in the points* x_i *and* y_j *yields the matrix* $X \in \mathbb{R}^{(n+m)\times(n+m)}$ *given by* $X_{ij} := g^{DLP}(x_i, y_j)$. X *is a typical block in the Nystrøm discretisation of the double layer potential operator on a cube in* $\mathbb{R}^3$. *The domains* Ω_{t_1} *and* Ω_{s_2} *as well as* Ω_{t_2} *and* Ω_{s_1} *lie in the same plane, while the outer normal vectors of* Ω_{t_1} *and* Ω_{s_1} *and* Ω_{t_2} *and* Ω_{s_2} *respectively are perpendicular. Thus, all entries* X_{ij} *with* $i \leq n$ *and* $j > n$ *or* $i > n$ *and* $j \leq n$ *vanish, while the diagonal blocks are non-zero. The matrix* X *bears the block structure*

$$X = \begin{pmatrix} X_{11} & 0 \\ 0 & X_{22} \end{pmatrix} \qquad X_{11} \in \mathbb{R}^{n\times n}, X_{22} \in \mathbb{R}^{m\times m}.$$

In [2, Algorithm 4.2], the first n row indices fulfil $1 \leq i_\nu^* \leq n$ for $\nu \in \{1, \ldots, n\}$. Due to the block structure, we can infer $1 \leq j_\nu^* \leq n$ for all $\nu \in \{1, \ldots, n\}$. For these pivot indices, the row and column vectors vanish outside of the block $X|_{n\times n}$ and the generated approximant AB^T is of the form

$$AB^T = \begin{pmatrix} A_1 \\ 0 \end{pmatrix} \begin{pmatrix} B_1 \\ 0 \end{pmatrix}^T = \begin{pmatrix} A_1 B_1^T & 0 \\ 0 & 0 \end{pmatrix}$$

with $A_1, B_1 \in \mathbb{R}^{n\times n}$. As a consequence, the approximation error satisfies

$$\|X - AB^T\|_2 \geq \|X_{22}\|_2,$$

which means that no convergence will occur until the rank exceeds n. In other words, the partially pivoted ACA algorithm works on the first block X_{11} (where we can expect it to converge exponentially) but does not recognise the second block X_{22}. If we chose the initial pivot index $i_1^* > n$, we would instead approximate the second block but not the first. In both cases, the error estimator $\|a_\mu\|_2\|b_\mu\|_2$ (as it is proposed in [2]) converges exponentially to zero, thus suggesting a good convergence of the total error, when, in fact, no convergence occurs.

Obviously, similar statements hold for collocation or Galerkin discretisations.

Example 2.2 implies that asymptotic smoothness of the kernel function in only *one* of the two variables does not ensure convergence of the ACA method presented in [2]. At least for our examples, there are heuristic modifications (cf. Algorithm 4) that seem to lead to good convergence.

We conclude that ACA works nicely for matrices X of the form $X_{ij} = g(x_i, y_j)$ where g is asymptotically smooth with respect to both variables, while it will possibly fail if this is not the case. Later, we will apply ACA to the matrix $S_{t,s}$ from (2.11) where these requirements are fulfilled.

2.3 Clustering and Standard $\mathcal{H}$-Matrix Compression

The low-rank approximations apply only to matrix blocks $t\times s \subset I\times I$ that are admissible with respect to (2.6). Therefore we have to subdivide the matrix into blocks that are admissible, preferably large blocks in order to compress the matrix by a maximal factor. The standard way to choose the blocks (testing all blocks would be too expensive and not lead to a useful partition) is to *cluster* the index set I hierarchically in a cluster tree T_I and use a canonical construction for the partition of the matrix.

Notation 2.3 (Tree) *Let $T = (V, E)$ be a tree with vertex set V and edge set E. The unique vertex $v \in V$ with $(w, v) \notin V$ for all $w \neq v$ is called the root of T and denoted by* $\operatorname{root}(T)$. *The levels of the tree are defined inductively by*

$$T^0 := \{\operatorname{root}(T)\}, \qquad T^{i+1} := \{w \in V \mid \exists v \in T^i : (v, w) \in E\}.$$

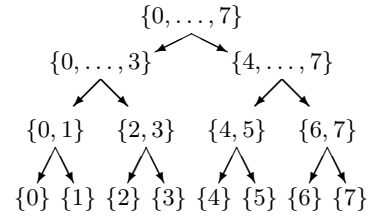
The set of successors of a node $v \in T^i$ is defined as $\operatorname{sons}(v) := \{w \in T^{i+1} \mid (v,w) \in E\}$. *The set of leaves (*$\operatorname{sons}(v) = \emptyset$*) is denoted by* $\mathcal{L}(T)$. *The depth of the tree T is given by*

$$p(T) := 1 + \max_{T^i \neq \emptyset} i.$$

We will use the short notation $v \in T$ instead of $v \in V$ and p instead of $p(T)$.

Definition 2.4 (Cluster Tree T_I) *A cluster tree T_I for an index set I is a tree with root* $\operatorname{root}(T_I) = I$ *where each vertex fulfils*

$$t = \dot{\bigcup_{s \in \operatorname{sons}(t)}} s \qquad \text{and } t \neq \emptyset.$$



For all practical cases the cluster tree T_I is a binary tree (each node has exactly two successors or it is a leaf), the depth p is $\mathcal{O}(\log \#I)$ and the cardinality of T_I is $\mathcal{O}(\#I)$. Next, we want to construct the cluster tree T_I with underlying index set $I = \{1, \ldots, n\}$. Each index $i \in I$ corresponds to one of the basis functions $\varphi_i \in V_n$ (cf. (2.5)). The geometric location of the index i is given by the Chebyshev centre x_i of the support of φ_i (the Chebyshev centre is the centre of the smallest ball containing the support of φ_i). Instead of the Chebyshev centre one could choose any point x_i from the support of φ_i.

Construction 2.5 (Geometrically Balanced Clustering) *We fix a bound $n_{\min} \in \mathbb{N}$ for the size of leaf clusters and construct the cluster tree T_I recursively. If a cluster t and a bounding box B satisfying $x_i \in B$ for all $i \in t$ are given, we introduce new bounding boxes B_1 and B_2 by splitting B in the coordinate direction of maximal extent and new son clusters by setting $t_1 := \{i \in t \ : \ x_i \in B_1\}$ and $t_2 := \{i \in t \ : \ x_i \in B_2\}$. If $\#t_1 > n_{\min}$ or $\#t_2 > n_{\min}$, we proceed by recursion.*

Construction 2.5 terminates if and only if the number of points x_i with the same geometric position is less than $n_{\min}$, and a minimal value of $n_{\min}$ guaranteeing termination can be determined a priori by an algorithm of complexity $\mathcal{O}(\#I)$.

Pairs of clusters $(t,s) \in T_I \times T_I$ are candidates for blocks of the matrix, and among these blocks we can choose those that we want to approximate in the $R(k)$-matrix format (cf. Definition 2.1). The number of all possible pairs is too large, therefore we test only pairs of clusters on the same level of the tree. This motivates the definition of a block cluster tree $T_{I\times I}$ that stores the admissible pairs of clusters in a hierarchical form.

Definition 2.6 (Block Cluster Tree $T_{I\times I}$) *A block cluster tree $T_{I\times I}$ based on the cluster tree T_I is a cluster tree for $I \times I$ such that (cf. Figure 2)*

$$\forall\, v \in T^i_{I\times I} \ \ \exists\, t,s \in T^i_I : \qquad v = t \times s.$$

Remark 2.7 (Cluster Tree yields Partition) *The leaves of a cluster tree T_I form a partition of the index set I. As a consequence, the leaves of a block cluster tree $T_{I\times I}$ yield a partition of the product index set $I \times I$.*

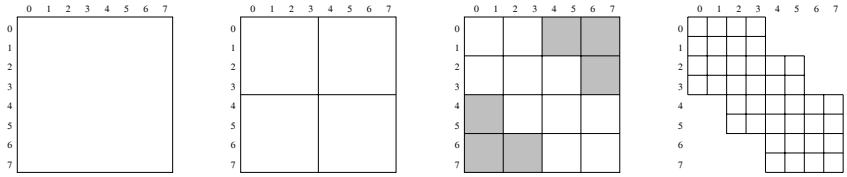


Figure 2: Depicted are four levels of a block cluster tree based on the cluster tree from Definition 2.4. On level 2 of the tree there are six leaves (shaded), e.g., the block $\{0,1\} \times \{6,7\}$.

If the underlying tree T_I is a binary tree, then the block cluster tree $T_{I\times I}$ is a quad-tree. Based on the cluster tree T_I (Construction 2.5) and the η-admissibility condition (2.6), we construct the canonical block cluster tree $T_{I\times I}$ as follows.

Construction 2.8 (Canonical Block Cluster Tree $T_{I\times I}$) *Let the cluster tree T_I be given. We define the block cluster tree $T_{I\times I}$ by* $\operatorname{root}(T) := I \times I$ *and for each vertex $t \times s \in T$ the set of successors by*

$$S(t\times s) := \begin{cases} \emptyset & \text{if } \min\{\#t, \#s\} \leq n_{\min} \\ \emptyset & \text{if } t\times s \text{ is } \eta\text{-admissible (2.6)} \\ S(t)\times S(s) & \text{otherwise.} \end{cases}$$

The block cluster tree $T_{I\times I}$ is the basis for the hierarchical matrix format. It defines the partition of the matrix into sub-matrices that are represented in the $R(k)$-matrix format.

Definition 2.9 ($\mathcal{H}$-Matrix) *Let $T := T_{I\times I}$ be a block cluster tree and $k : \mathcal{L}(T) \to \mathbb{N}_0$ a rank distribution. We define the set $\mathcal{H}(T,k)$ of hierarchical matrices ($\mathcal{H}$-matrices) by*

$$\mathcal{H}(T,k) := \left\{ X \in \mathbb{R}^{I\times I} \;\middle|\; \forall t\times s \in \mathcal{L}(T) : \operatorname{rank}(X|_{t\times s}) \leq k(t\times s) \right\}.$$

A matrix $X \in \mathcal{H}(T,k)$ is said to be given in $\mathcal{H}$-matrix representation, if for all leaves $t \times s$ with $\#t \leq n_{\min}$ or $\#s \leq n_{\min}$ the corresponding matrix block $X|_{t\times s}$ is given in full matrix representation and in $R(k)$-matrix representation for the other leaves.

The result of Construction 2.5 followed by Construction 2.8 is a block cluster tree $T_{I\times I}$ for which we can estimate the depth and the sparsity C_{sp} defined next. The sparsity is needed to estimate the storage requirements and complexity of the matrix-vector multiplication of an $\mathcal{H}$-matrix.

Definition 2.10 (Sparsity) *Let $T_{I\times I}$ be a block cluster tree. We define the* sparsity (constant) C_{sp} *of $T_{I\times I}$ by*

$$C_{\text{sp}} := \max_{t\in T_I} \#\{s \in T_I \mid t\times s \in T_{I\times I}\}.$$

So far, we have not imposed any condition on the locality of the supports of the basis functions φ_i. If all the supports cover the whole domain Ω, then there is no admissible block, so we have to demand locality of the supports in order to be able to apply our technique.

Assumption 2.11 (Locality) *We assume that the supports are locally separated in the sense that there exist two constants C_{sep} and $n_{\min}$ such that*

$$\max_{i\in I} \#\{j \in I \mid \operatorname{dist}(\operatorname{supp}\varphi_i, \operatorname{supp}\varphi_j) \leq C_{\mathrm{sep}}^{-1} \operatorname{diam}(\operatorname{supp}\varphi_i)\} \leq n_{\min}. \tag{2.12}$$

The left-hand side is the maximal number of basis functions with 'relatively close' supports (see Figure 3). Note that we will apply Construction 2.5 with a parameter $n_{\min}$ that satisfies (2.12).

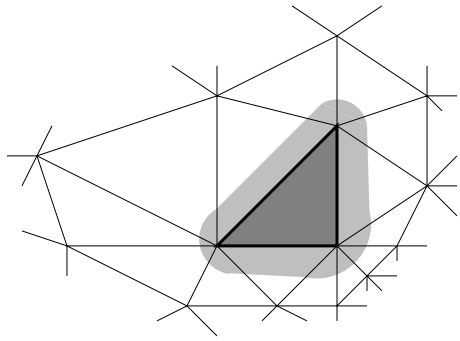


Figure 3: The triangle $\Omega_i := \operatorname{supp}\varphi_i$ under consideration is dark grey. The area with a distance of $C_{\mathrm{sep}}^{-1}\operatorname{diam}(\Omega_i)$ is light grey ($C_{\mathrm{sep}} := 4\sqrt{2}$). Here, 15 triangles (including Ω_i) are 'rather close' to Ω_i.

Based upon Assumption 2.11 the sparsity constant C_{sp} as well as the depth p of the block cluster tree $T_{I\times I}$ is estimated in [15, Lemma 4.5]. It should be noted that $C_{\mathrm{sp}} = \mathcal{O}(\eta^{-3})$ and thus the choice of η in the admissibility condition (2.6) enters the estimate in a critical way.

The estimates for the storage requirements and the complexity of the matrix-vector multiplication of an $\mathcal{H}$-matrix depend only on the cardinality of I and the depth and sparsity of the block cluster tree T:

Lemma 2.12 (Storage) *Let T be a block cluster tree based on the cluster tree T_I with sparsity constant C_{sp} (Definition 2.10) and minimal block size $n_{\min}$. Then the storage requirements $N_{\mathcal{H},St}(T,k)$ for an $\mathcal{H}$-matrix $X \in \mathcal{H}(T,k)$ are bounded by*

$$N_{\mathcal{H},St}(T,k) \leq 2(1+\operatorname{depth}(T))C_{\mathrm{sp}} \max\{k, n_{\min}\}\#I.$$

Proof. [15, Lemma 2.4] ∎

	Sphere			Cube		
	Time	Strg.	Rel. Err.	Time	Strg.	Rel. Err.
ACA, $\varepsilon = 10^{-2}$	238	9.2	3.3×10^{-3}	474	8.1	4.7×10^{-2}
ACA, $\varepsilon = 10^{-3}$	281	11.2	7.5×10^{-4}	553	10.7	4.4×10^{-2}
ACA, $\varepsilon = 10^{-4}$	350	14.2	3.3×10^{-5}	658	13.9	4.2×10^{-2}
ACA, $\varepsilon = 10^{-5}$	419	17.1	3.5×10^{-6}	771	17.2	4.2×10^{-2}
Interpol., $m = 2$	267	44.1	1.3×10^{-2}	282	39.1	6.5×10^{-2}
Interpol., $m = 3$	561	82.3	6.0×10^{-3}	568	74.8	5.2×10^{-3}
Interpol., $m = 4$	1183	124.5	5.2×10^{-4}	1164	127.6	8.1×10^{-4}
Interpol., $m = 5$	2426	175.9	7.0×10^{-5}	2082	178.8	5.0×10^{-5}

Table 1: Comparison of ACA and tensor interpolation on the unit sphere and the unit cube (without further recompression)

Lemma 2.13 (Matrix-Vector Product) *Let T be a block cluster tree. The complexity $N_{\mathcal{H}\cdot v}(T,k)$ of the matrix-vector product in the set of $\mathcal{H}$-matrices can be bounded from above and below by*

$$N_{\mathcal{H},St}(T,k) \leq N_{\mathcal{H}\cdot v}(T,k) \leq 2N_{\mathcal{H},St}(T,k).$$

2.4 Numerical Example: ACA versus Interpolation

In order to compare the algebraic method ACA and the analytic tensor interpolation we apply the two methods to two different geometries Ω, first the unit sphere and second the unit cube. We consider the model problem from Section 2.1 with the kernel function g^{DLP} of the double layer potential. The two domains are discretised with $n_{\mathrm{cube}} = 30000$ and $n_{\mathrm{sphere}} = 20000$ panels and basis functions φ_i that are constant on each panel. The minimal block size is set to $n_{\min} := 20$ and the admissibility parameter is $\eta := 2$. The parameter ε in the stopping criterion for ACA is chosen as 10^{-j} for $j = 2, \ldots, 5$.

The results in Table 1 show that ACA is advantageous for the sphere (less storage requirements and shorter setup time), but also that the method fails to reduce the relative inversion error below 10^{-2} in the case of the unit cube (compare Example 2.2). The column labeled "Time" gives the time in seconds for constructing the approximation, the column "Strg." gives the storage required for the resulting $\mathcal{H}$-matrix in kilobytes per degree of freedom (keep in mind that this amount will be reduced by coarsening strategies) and the column "Rel. Err." gives the relative error $\|I - \widetilde{G}^{-1}G\|_2$ approximated using a power iteration.

It should be noted that the storage requirements are not an immediate difficulty, since a simple algebraic recompression [14] eliminates the unnecessary singular vectors and coarsens the block structure automatically. However, the initial rank generated by either ACA or tensor interpolation enters the complexity quadratically, so that the time for the coarsening will be increased. For all computed examples the time for the coarsening is less than for the assembly.

The results in Table 1 were obtained by the HLib package [4] on one 900 MHz UltraSPARC IIIcu processor in a SunFire 6800 computer.

In the following section we present a combined method, the hybrid cross approximation (HCA) technique, that inherits the provable approximation property from the tensor interpolation while keeping the computational complexity close to that of the ACA heuristic.

3 Hybrid Cross Approximation

3.1 First Approach: Lagrange Polynomials

Since the ACA algorithm works well for pointwise evaluations of an asymptotically smooth kernel function, we will apply it to the coupling matrix $S_{t,s}$ from (2.11). The matrices U_t and V_s will neither be computed nor stored, instead we exploit the fact that they are nested.

Let t be a cluster with a son t'. Since we use interpolation of constant order, we have

$$\operatorname{span}\{\mathcal{L}_\nu^t \ : \ \nu \in \{1,\ldots,M\}\} = \operatorname{span}\{\mathcal{L}_{\nu'}^{t'} \ : \ \nu' \in \{1,\ldots,M\}\},$$

i.e., the different Lagrange bases span the same polynomial space. This implies

$$\mathcal{L}_\nu^t = \sum_{\nu'=1}^{M} \mathcal{L}_\nu^t(x_{\nu'}^{t'})\mathcal{L}_{\nu'}^{t'}.$$

We collect the coefficients of the basis transformation in a *transfer matrix* $T_{t'}$ defined by

$$(T_{t'})_{\nu'\nu} := \mathcal{L}_\nu^t(x_{\nu'}^{t'}).$$

For $i \in t'$, we find

$$\begin{aligned}(U_t)_{i\nu} &= \int_{\Omega^t} \phi_i(x)\mathcal{L}_\nu^t(x)\,\mathrm{d}x \\ &= \sum_{\nu'=1}^{M} (T_{t'})_{\nu'\nu} \int_{\Omega^t} \phi_i(x)\mathcal{L}_{\nu'}^{t'}(x)\,\mathrm{d}x = (U_{t'}T_{t'})_{i\nu}.\end{aligned}$$

Obviously, the same relationship holds for the matrices V_s, so we have to compute U_t and V_s only for leaves of the cluster tree and can construct all remaining matrices by using the transfer matrices.

We are now able to formulate the hybrid cross approximation based on tensor interpolation. This first variant HCA(I) (cf. Algorithm 2) is closer to interpolation than ACA, in particular one could replace ACA by any rank revealing scheme.

In order to set up the matrices U_t and V_s, we have to compute $(\#t+\#s)M$ integrals of the form $\int_\Omega \phi_i(x)\mathcal{L}_\nu^t(x)\mathrm{d}x$. Since we use nested cluster bases, we have to compute these integrals only for the leaves of the cluster tree, and since the leaves form a partition of I, we only have to evaluate $2M\#I$ integrals.

Algorithm 2 HCA(I)

procedure HCA1($S_{t,s}$, **var** A, B)
Call ACA($S_{t,s}$, $\widehat{A}$, $\widehat{B}$) with $\widehat{A}, \widehat{B} \in \mathbb{R}^{M \times k}$ so that

$$\|S_{t,s} - \widehat{A}\widehat{B}^T\|_2 \leq \varepsilon \|S_{t,s}\|_2$$

Multiply $A := U_t \widehat{A}$ and $B := V_s \widehat{B}$ using the backward transformation.

A further orthogonalisation [3] of the cluster basis can be done on-the-fly and reduces the storage requirements for the basis. The multiplications $U_t A$, $V_t B$ in the nested structure can be performed efficiently by using the $\mathcal{H}^2$-matrix backward transformation [7, 8]. Instead of the ACA algorithm with complexity $\mathcal{O}(k^2 M)$ one could as well compute an SVD of $S_{t,s}$ in complexity $\mathcal{O}(M^3)$.

3.2 Numerical Example for HCA(I)

We consider the model problems from Section 2.4, namely the Galerkin discretisation of the double layer potential operator on the unit sphere and unit cube by piecewise constant basis functions on a quasi-uniform grid.

			Sphere			Cube		
	ε	m	Time	Strg.	Rel. Err.	Time	Strg.	Rel. Err.
ACA	10^{-2}		238	9.2	3.3×10^{-3}	474	8.1	4.7×10^{-2}
ACA	10^{-3}		281	11.2	7.5×10^{-4}	553	10.7	4.4×10^{-2}
ACA	10^{-4}		350	14.2	3.3×10^{-5}	658	13.9	4.2×10^{-2}
ACA	10^{-5}		419	17.1	3.5×10^{-6}	771	17.2	4.2×10^{-2}
HCA(I)	10^{-3}	2	205	16.2	1.6×10^{-2}	229	13.3	6.3×10^{-2}
HCA(I)	10^{-4}	3	338	21.6	3.3×10^{-3}	372	18.0	1.8×10^{-2}
HCA(I)	10^{-5}	4	619	27.5	3.5×10^{-4}	609	25.5	3.1×10^{-3}
HCA(I)	10^{-6}	5	1135	33.9	1.4×10^{-4}	1170	29.4	2.7×10^{-4}

Table 2: Comparison of ACA and HCA(I)

Table 2 contains the results of a comparison between the ACA heuristic and the interpolation-based HCA(I) algorithm. The column labeled "Time" gives the time in seconds for constructing the approximation, the column "Strg." gives the storage required for the resulting $\mathcal{H}$-matrix in kilobytes per degree of freedom (keep in mind that this amount will be reduced by coarsening strategies) and the column "Rel. Err." gives the relative error $\|I - \widetilde{G}^{-1}G\|_2$ approximated using a power iteration.

We can see that HCA(I) requires more time than the ACA heuristic, but also that it converges on the unit cube while ACA fails.

3.3 Second Approach: Cross Approximation

The idea for our second approach is to approximate the generator function $\gamma(x,y)$ in an admissible bounding box $B_t \times B_s$ by some functional skeleton

$$\tilde{\gamma}_1(x,y) = \gamma(x,y_{j_1})\gamma(x_{i_1},y)/\gamma(x_{i_1},y_{j_1}),$$

where x_{i_1} and y_{j_1} are appropriate interpolation points from an mth order interpolation scheme in $B_t \times B_s$. The pivot elements i_ℓ and j_ℓ coincide with those from an ACA approximation of $S_{t,s}$, because a rank 1 ACA approximation is just the evaluation of $\tilde{\gamma}_1$ in $(x_i,y_j)_{i,j=1}^M$.

Successively, we approximate the remainder $\gamma - \sum_{\ell=1}^{k-1}\tilde{\gamma}_\ell$ in the same way and obtain in the end an approximation of the form

$$\tilde{\gamma}(x,y) := \sum_{\ell=1}^{k}\left(\sum_{q=1}^{\ell}\gamma(x,y_{j_q})C_{\ell,q}\right)\left(\sum_{q=1}^{\ell}\gamma(x_{i_q},y)D_{\ell,q}\right),$$

where $C_{\ell,q}, D_{\ell,q}$ are given by recursion formulae. The final degenerate kernel is defined by

$$\begin{aligned}\tilde{g} &:= D_x D_y \tilde{\gamma} \\ &= \sum_{\ell=1}^{k}\left(\sum_{q=1}^{\ell}D_x\gamma(x,y_{j_q})C_{\ell,q}\right)\left(\sum_{q=1}^{\ell}D_y\gamma(x_{i_q},y)D_{\ell,q}\right).\end{aligned}$$

The twofold integrals $\int_\Omega\int_\Omega \phi_i(x)g(x,y)\phi_j(y)$ will now resolve into single integrals of the form

$$\int_\Omega \phi_i(x)D_x\gamma(x,y_{j_\ell})\mathrm{d}x, \quad \int_\Omega \phi_j(y)D_y\gamma(x_{i_\ell},y)\mathrm{d}y,$$

and thus the complexity of a q-point quadrature per basis function in the Galerkin discretisation is reduced from squared cost $((\#t+\#s)kq^2)$ to linear cost $((\#t+\#s)kq)$. For collocation discretisations the integration with respect to one of the two variables is replaced by a simple evaluation of the kernel.

In order to compute the k^2 entries of C, D in Algorithm 3, we have to perform $\mathcal{O}(k^3)$ arithmetical operations. The total complexity for HCA(II) amounts to $\mathcal{O}((\#t+\#s)k^2 + Mk^2)$.

3.4 Numerical Example for HCA(II)

Again, we consider the model problem from Section 2.4. Table 3 contains the results of a comparison between the ACA heuristic and the cross-approximation based HCA(II) algorithm. The column labeled "Time" gives the time in seconds for constructing the approximation, the column "Strg." gives the storage required for the resulting $\mathcal{H}$-matrix in kilobytes per degree of freedom (keep in mind that this amount will be reduced by

Algorithm 3 HCA(II)

procedure HCA2($S_{t,s}$, **var** A, B)
Call ACA($S_{t,s}$, $\widehat{A}$, $\widehat{B}$) with $\widehat{A}, \widehat{B} \in \mathbb{R}^{M\times k}$ so that

$$\|S_{t,s} - \widehat{A}\widehat{B}^T\|_2 \leq \varepsilon \|S_{t,s}\|_2$$

and store the pivot indices $(i_\ell)_{\ell=1}^k, (j_\ell)_{\ell=1}^k$
Initialise $C, D \in \mathbb{R}^{k\times k}$ and $c, d \in \mathbb{R}^k$ by zero
for $\ell = 1, \ldots, k$ **do**
  for $i = 1, \ldots, \ell - 1$ **do**
    $d_i := 0, c_i := 0$
    for $q = 1, \ldots, i$ **do**
      $c_i := c_i + C_{i,q}\gamma(x_{i_\ell}, y_{j_q})$
      $d_i := d_i + D_{i,q}\gamma(x_{i_q}, y_{j_\ell})$
    end for
  end for
  $C_{\ell,\ell} := 1/\sqrt{|(\hat{a}_\ell)_{i_\ell}|}$, $D_{\ell,\ell} := \operatorname{sign}((\hat{a}_\ell)_{i_\ell})/\sqrt{|(\hat{a}_\ell)_{i_\ell}|}$
  for $q = 1, \ldots, \ell - 1$ **do**
    $C_{\ell,q} := 0$, $D_{\ell,q} := 0$
    for $i = q, \ldots, \ell - 1$ **do**
      $C_{\ell,q} := C_{\ell,q} - C_{i,q} d_i C_{\ell,\ell}$
      $D_{\ell,q} := D_{\ell,q} - D_{i,q} c_i D_{\ell,\ell}$
    end for
  end for
end for
Compute the entries of $\widehat{U} \in \mathbb{R}^{t\times k}$ and $\widehat{V} \in \mathbb{R}^{s\times k}$ by

$$\widehat{U}_{i\ell} := \int_{\Omega_t} \phi_i(x) D_x\gamma(x, y_{j_\ell}), \quad \widehat{V}_{j\ell} := \int_{\Omega_s} \phi_j(y) D_y\gamma(x_{i_\ell}, y)$$

Multiply $A := \widehat{U}C^T$ and $B := \widehat{V}D^T$

coarsening strategies) and the column "Rel. Err." gives the relative error $\|I - \widetilde{G}^{-1}G\|_2$ approximated using a power iteration.

We can see that HCA(II) requires roughly the same time as the ACA heuristic, but additionally it converges on the unit cube while ACA fails.

			Sphere			Cube		
	ε	m	Time	Strg.	Rel. Err.	Time	Strg.	Rel. Err.
ACA	10^{-2}		238	9.2	3.3×10^{-3}	474	8.1	4.7×10^{-2}
ACA	10^{-3}		281	11.2	7.5×10^{-4}	553	10.7	4.4×10^{-2}
ACA	10^{-4}		350	14.2	3.3×10^{-5}	658	13.9	4.2×10^{-2}
ACA	10^{-5}		419	17.1	3.5×10^{-6}	771	17.2	4.2×10^{-2}
HCA(II)	10^{-3}	2	213	17.1	3.4×10^{-3}	246	14.4	1.5×10^{-2}
HCA(II)	10^{-4}	3	275	24.8	9.5×10^{-4}	326	21.2	2.9×10^{-3}
HCA(II)	10^{-5}	4	368	32.5	9.9×10^{-5}	451	28.2	6.8×10^{-4}
HCA(II)	10^{-6}	5	494	40.5	4.2×10^{-6}	580	36.8	1.5×10^{-4}

Table 3: Comparison of ACA and HCA(II)

4 Analysis

4.1 Interpolation Error

Let $I_m : C[-1,1] \to \mathcal{P}_m$ for $m \in \mathbb{N}$ be a stable interpolation operator of order m satisfying the projection property

$$I_m[p] = p \qquad \text{for all } p \in \mathcal{P}_m \tag{4.1}$$

and the stability property

$$\|I_m[f]\|_\infty \leq \Lambda_m \|f\|_\infty \qquad \text{for all } f \in C[a,b]. \tag{4.2}$$

For a given set $(x_{m,\nu})_{\nu=0}^m$ of interpolation points, the operator I_m is of the form

$$I_m[f](x) = \sum_{\nu=0}^{m} f(x_{m,\nu}) \mathcal{L}_{m,\nu}(x),$$

where the Lagrange polynomials are given by

$$\mathcal{L}_{m,\nu}(x) := \prod_{\mu=0,\mu\neq\nu}^{m} \frac{x - x_\mu}{x_\nu - x_\mu}.$$

The Chebyshev interpolation operator is defined by setting $x_{m,\nu} = \cos(\pi(\nu+1)/(2m+2))$ and for this operator the stability constant is bounded by $\Lambda_m \leq (2/\pi)\ln(m+1)+1 \leq m+1$ (cf. [20]).

Lemma 4.1 (Stability of derivatives) *The estimate*

$$\|(I_m[f])'\|_\infty \leq \Lambda_m m^2 \|f'\|_\infty$$

holds for all $f \in C^1[-1,1]$.

Proof. Let $f_0 \equiv f(0)$. Due to $f_0' = 0$ and $I_m[f_0] = f_0$ (cf. (4.1)), we find

$$\|(I_m[f])'\|_\infty = \|(I_m[f] - f_0)'\|_\infty = \|(I_m[f - f_0])'\|_\infty.$$

Markov's inequality [10, Theorem 4.1.4] implies

$$\|(I_m[f - f_0])'\|_\infty \leq m^2 \|I_m[f - f_0]\|_\infty,$$

and we can use the stability (4.2) of I_m to find

$$\|I_m[f - f_0]\|_\infty \leq \Lambda_m \|f - f_0\|_\infty.$$

By the standard error estimate for the zeroth-order Taylor expansion, we get

$$\|f - f_0\|_\infty \leq \|f'\|_\infty$$

and can conclude

$$\|(I_m[f])'\|_\infty \leq \Lambda_m m^2 \|f'\|_\infty.$$

This is the desired estimate (note that this proof can be generalized for higher derivatives by replacing f_0 by appropriate Taylor polynomials). ∎

The multi-dimensional case can be treated by tensor arguments. Let $d \in \mathbb{N}$ and $B := J^1 \times \ldots \times J^d$ for closed intervals $J^1, \ldots, J^d$ of positive length. For each $i \in \{1, \ldots, d\}$, we fix the (unique) affine monotonous bijective map $\Phi^i : [-1, 1] \to J^i$ and define the transformed interpolation operator

$$I_m^i : C(J^i) \to \mathcal{P}_m, \quad f \mapsto (I_m^i[f \circ \Phi^i]) \circ (\Phi^i)^{-1}.$$

Lemma 4.1 also holds for the operators I_m^i, $i \in \{1, \ldots, d\}$. The tensor interpolation operator for the box B is given by

$$I_m^B := I_m^1 \otimes \ldots \otimes I_m^d$$

and we can prove the following generalisation of the result of [5, Lemma 2.1]:

Lemma 4.2 (Multidimensional Stability) *Let $\alpha \in \{0, 1\}^d$. We have*

$$\|\partial^\alpha I_m^B[f]\|_\infty \leq \Lambda_m^d m^{2|\alpha|} \|\partial^\alpha f\|_\infty$$

for all $f \in C^1(B)$.

Proof. Let $f \in C^1(B)$ and $k, \ell \in \{1, \ldots, d\}$. We denote the interpolation points and Lagrange polynomials corresponding to J^ℓ by

$$x_{m,\nu}^\ell := \Phi^\ell(x_{m,\nu}), \quad \mathcal{L}_{m,\nu}^\ell := \mathcal{L}_{m,\nu} \circ (\Phi^\ell)^{-1}$$

and define the interpolation operator in the ℓth component by

$$I_m^{B,\ell} := (I \otimes \ldots \otimes I \otimes I_m^\ell \otimes I \otimes \ldots \otimes I).$$

We observe

$$I_m^{B,\ell}[f](x) = \sum_{\nu=0}^{m} f(x_1, \dots, x_{\ell-1}, x_{m,\nu}^{\ell}, x_{\ell+1}, \dots, x_d) \mathcal{L}_{m,\nu}^{\ell}(x_\ell).$$

Differentiating and using (4.2) and Lemma 4.1 yields

$$\left\| \partial_k (I_m^{B,\ell}[f]) \right\|_\infty \leq \begin{cases} \Lambda_m m^2 \|\partial_k f\|_\infty & \text{if } \ell = k \\ \Lambda_m \|\partial_k f\|_\infty & \text{otherwise.} \end{cases}$$

Due to

$$I_m^B = \prod_{\ell=1}^{d} I_m^{B,\ell},$$

this inequality implies the desired estimate. ∎

In order to prove an error estimate for the derivatives of multi-dimensional tensor interpolations, we will apply [5, Theorem 3.2] to a suitable derivative of γ. Since the error estimate only holds for asymptotically smooth functions, we have to demonstrate that derivatives of asymptotically smooth functions are again asymptotically smooth.

Lemma 4.3 (Smoothness of Derivatives) *Let $\alpha \in \{0,1\}^d$, let $f \in C^\infty(B)$ such that there are constants $C_0, c_0 \in \mathbb{R}_{\geq 1}$ satisfying*

$$\|\partial^\beta f\|_\infty \leq C_0 c_0^{|\beta|} \beta!$$

for all $\beta \in \mathbb{N}_0^d$. Let $c_1 \in \mathbb{R}_{>c_0}$. There is a constant $C \in \mathbb{R}$, depending only on α, d and c_0/c_1, such that

$$\|\partial^\beta \partial^\alpha f\|_\infty \leq C_1 c_1^{|\beta|} \beta!$$

holds with $C_1 = C C_0 c_0^{|\alpha|}$ for all $\beta \in \mathbb{N}_0^d$.

Proof. We find

$$\begin{aligned} \|\partial^\beta \partial^\alpha f\|_\infty &= \|\partial^{\alpha+\beta} f\|_\infty \\ &\leq C_0 c_0^{|\alpha|} c_0^{|\beta|} (\alpha+\beta)! = C_0 c_0^{|\alpha|} \left(\prod_{i=1,\alpha_i \neq 0}^{d} (\beta+1) \right) c_0^{|\beta|} \beta! \\ &\leq C_0 c_0^{|\alpha|} \left(\prod_{i=1,\alpha_i \neq 0}^{d} (\beta+1) \right) \left(\frac{c_0}{c_1} \right)^{|\beta|} c_1^{|\beta|} \beta!. \end{aligned}$$

Due to $c_1 > c_0$, we have $c_0/c_1 < 1$ and can find a constant $C \in \mathbb{R}_{>0}$ satisfying

$$\left(\prod_{i=1,\alpha_i \neq 0}^{d} (\beta+1) \right) \left(\frac{c_0}{c_1} \right)^{|\beta|} \leq C$$

for all $\beta \in \mathbb{N}_0^d$. The proof closes by setting $C_1 := CC_0c_0^{|\alpha|}$ and observing

$$\begin{aligned}\|\partial^\beta \partial^\alpha f\|_\infty &\leq C_0 c_0^{|\alpha|} \left(\prod_{i=1,\alpha_i\neq 0}^{d} (\beta+1) \right) \left(\frac{c_0}{c_1}\right)^{|\beta|} c_1^{|\beta|}\beta! \\ &\leq C_0 c_0^{|\alpha|} C c_1^{|\beta|}\beta! = C_1 c_1^{|\beta|}\beta!.\end{aligned}$$

■

Now we can proceed to prove an error estimate for the derivatives of multi-dimensional tensor interpolants by generalising the result of [5, Theorem 3.2]:

Theorem 4.4 (Approximation of Derivatives) *Let $\alpha, \beta \in \{0,1\}^3$. Let $m \geq 1$. Let $c_1 > c_0$. There is a polynomial C_{apx} satisfying*

$$\|\partial_x^\alpha \partial_y^\beta (\gamma - I_m^{B_t \times B_s}[\gamma])\|_\infty \leq \frac{C_{\mathrm{apx}}(m)\Lambda_m^6 \Lambda_{m-1}^6}{(C_{\mathrm{as2}} \operatorname{dist}(B_t, B_s))^{\sigma+|\alpha+\beta|}} \left(\frac{\eta}{\eta + C_{\mathrm{as2}}}\right)^m.$$

for all $m \in \mathbb{N}$ and all admissible bounding boxes B_t and B_s.

Proof. Let $B := B_t \times B_s$, let $d = 6$, let $\mu := (\alpha, \beta) \in \mathbb{N}_0^d$. The generator function γ is asymptotically smooth with

$$C_0 = C_{\mathrm{as1}}(C_{\mathrm{as2}} \operatorname{dist}(B_t, B_s))^{-\sigma} \quad \text{and} \quad c_0 := C_{\mathrm{as2}}^{-1},$$

and this implies $\gamma|_B \in C^\infty(B)$. Due to Lemma 4.3, the function $\gamma' := \partial_x^\alpha \partial_y^\beta \gamma = \partial^\mu \gamma$ is also asymptotically smooth with

$$C_1 = C(C_{\mathrm{as2}} \operatorname{dist}(B_t, B_s))^{-\sigma-|\alpha|} \quad \text{and} \quad c_1 := \sqrt{2}c_0$$

for a constant C that does not depend on B_t, B_s or m. We can apply [5, Theorem 3.2] in order to find a tensor polynomial $\tilde{\gamma}'$ of order $m-1$ and a constant $\tilde{C} := 8e$ satisfying

$$\|\gamma' - \tilde{\gamma}'\|_\infty \leq \frac{\tilde{C}(1 + c_1 \operatorname{diam}(B_t \times B_s))C_1 \Lambda_{m-1}^d m}{\left(1 + \frac{2}{c_1 \operatorname{diam}(B_t \times B_s)}\right)^m}. \tag{4.3}$$

Since B_t and B_s are admissible, (2.6) holds and we find

$$\begin{aligned} c_1 \operatorname{diam}(B_t \times B_s) &\leq c_1 (\operatorname{diam}(B_t)^2 + \operatorname{diam}(B_s)^2)^{1/2} \\ &\leq c_1 \sqrt{2} \max\{\operatorname{diam}(B_t), \operatorname{diam}(B_s)\} \\ &\leq 2c_0 \eta \operatorname{dist}(B_t, B_s) = \frac{2\eta \operatorname{dist}(B_t, B_s)}{C_{\mathrm{as2}} \operatorname{dist}(B_t, B_s)} \\ &= 2\eta / C_{\mathrm{as2}}, \end{aligned}$$

so the error estimate (4.3) takes the form

$$\begin{aligned}\|\gamma' - \tilde{\gamma}'\|_\infty &\leq \frac{C\tilde{C}(1 + 2\eta/C_{\mathrm{as2}})\Lambda_{m-1}^d m}{(C_{\mathrm{as2}} \operatorname{dist}(B_t, B_s))^{\sigma+|\mu|}} \left(1 + \frac{C_{\mathrm{as2}}}{\eta}\right)^{-m} \\ &= \frac{C'(m)\Lambda_{m-1}^d}{(C_{\mathrm{as2}} \operatorname{dist}(B_t, B_s))^{\sigma+|\mu|}} \left(\frac{\eta}{\eta + C_{\mathrm{as2}}}\right)^m\end{aligned}$$

with $C'(m) := C\tilde{C}(1 + 2\eta/C_{\mathrm{as2}})m$.

Let $\tilde{\gamma}$ be an antiderivative of $\tilde{\gamma}'$ satisfying

$$\partial_x^\alpha \partial_y^\beta \tilde{\gamma} = \tilde{\gamma}'.$$

Since $\tilde{\gamma}'$ is a tensor polynomial of order $m-1$, $\tilde{\gamma}$ is a tensor polynomial of order m and we can apply Lemma 4.1 to find

$$\begin{aligned}
\|\partial_x^\alpha \partial_y^\beta \gamma - \partial_x^\alpha \partial_y^\beta I_m^B[\gamma]\|_\infty &\leq \|\partial_x^\alpha \partial_y^\beta \gamma - \partial_x^\alpha \partial_y^\beta \tilde{\gamma}\|_\infty + \|\partial_x^\alpha \partial_y^\beta I_m^B[\gamma - \tilde{\gamma}]\|_\infty \\
&\leq (1 + \Lambda_m^d m^{2|\alpha+\beta|})\|\partial_x^\alpha \partial_y^\beta (\gamma - \tilde{\gamma})\|_\infty \\
&= (1 + \Lambda_m^d m^{2|\alpha+\beta|})\|\gamma' - \tilde{\gamma}'\|_\infty \\
&\leq \frac{C'(m)\Lambda_{m-1}^d \Lambda_m^d (1 + m^{2|\alpha+\beta|})}{(C_{\mathrm{as2}} \operatorname{dist}(B_t, B_s))^{\sigma+|\alpha+\beta|}} \left(\frac{\eta}{\eta + C_{\mathrm{as2}}}\right)^m.
\end{aligned}$$

Setting $C_{\mathrm{apx}}(m) := C'(m)(1 + m^{2|\alpha+\beta|})$ concludes the proof. ∎

In order to find an error estimate for $\tilde{g}$, we have to bound the norms of the coefficients. We set

$$\begin{aligned}
\|c_0^x\| &:= \sup\{|c_0^x(x)| \ : \ x \in \Omega\}, \\
\|c^x\| &:= \sup\{|c_1^x(x)| + |c_2^x(x)| + |c_3^x(x)| \ : \ x \in \Omega\}, \\
\|c_0^y\| &:= \sup\{|c_0^y(y)| \ : \ y \in \Omega\}, \\
\|c^y\| &:= \sup\{|c_1^y(y)| + |c_2^y(y)| + |c_3^y(y)| \ : \ x \in \Omega\}
\end{aligned}$$

and get the following error bound:

Corollary 4.5 (Separable Approximation) *There is a polynomial C_g such that*

$$\begin{aligned}
\|g - \tilde{g}\|_{\infty,\Omega_t\times\Omega_s} \leq \frac{C_g(m)\Lambda_m^6 \Lambda_{m-1}^6}{(C_{\mathrm{as2}} \operatorname{dist}(B_t, B_s))^\sigma} \left(\frac{\eta}{\eta + C_{\mathrm{as2}}}\right)^m \\
\left(\|c_0^x\| + \frac{C_{\mathrm{as2}}^{-1}\|c^x\|}{\operatorname{dist}(B_t, B_s)}\right)\left(\|c_0^y\| + \frac{C_{\mathrm{as2}}^{-1}\|c^y\|}{\operatorname{dist}(B_t, B_s)}\right)
\end{aligned}$$

holds for all $m \in \mathbb{N}$ and all admissible bounding boxes B_t, B_s.

Proof. This is a direct consequence of Theorem 4.4. ∎

We will now use this estimate to derive error bounds for the discrete matrices. In order to do so, we fix a constant $C_{\mathrm{fe}} \in \mathbb{R}_{>0}$ satisfying

$$\|\sum_{i\in I} x_i \varphi_i\|_{L^2}^2 \leq C_{\mathrm{fe}} h^{d_\Omega} \|x\|_2^2 \tag{4.4}$$

for all vectors $x \in \mathbb{R}^I$, a mesh size $h \in \mathbb{R}_{>0}$ and the intrinsic dimension $d_\Omega \in \mathbb{N}$ of Ω.

Lemma 4.6 (Low-rank Approximation) *Let $t \times s$ be an admissible block, and let $\tilde{g}$ be an approximation of g. Let $C_{\mathrm{fe}}, d_\Omega, h$ be as in (4.4). Then the factorized matrix from (2.9) satisfies*

$$\|G|_{t\times s} - U_t S_{t,s} V_s^T\|_2 \leq C_{\mathrm{fe}} |\Omega_t|^{1/2} |\Omega_s|^{1/2} h^{d_\Omega} \|g - \tilde{g}\|_{\infty,\Omega_t\times\Omega_s}.$$

Proof. Let $u \in \mathbb{R}^t$ and $v \in \mathbb{R}^s$. Let

$$\hat{u} := \sum_{i\in t} u_i \varphi_i, \qquad \hat{v} := \sum_{j\in t} v_j \varphi_j.$$

Using the short notation $\delta := \|g - \tilde{g}\|_{\infty,\Omega_t\times\Omega_s}$ we find

$$\begin{aligned}
|\langle u, (G|_{t\times s} - U_t S_{t,s} V_s^T) v\rangle| &= \left| \int_{\Omega_t} \int_{\Omega_s} (g - \tilde{g})(x,y) \hat{u}(x) \hat{v}(y) \,\mathrm{d}y \,\mathrm{d}x \right| \\
&\leq \delta \int_{\Omega_t} |\hat{u}(x)| \,\mathrm{d}x \int_{\Omega_s} |\hat{v}(y)| \,\mathrm{d}y \\
&\overset{C-S}{\leq} \delta |\Omega_t|^{1/2} \|\hat{u}\|_{L^2} |\Omega_s|^{1/2} \|\hat{v}\|_{L^2} \\
&\overset{(4.4)}{\leq} C_{\mathrm{fe}} \delta |\Omega_t|^{1/2} |\Omega_s|^{1/2} h^{d_\Omega} \|u\|_2 \|v\|_2.
\end{aligned}$$

Since this estimate holds for arbitrary u and v, it implies the desired upper bound. ∎

Corollary 4.7 (Chebyshev Interpolation) *Let $t\times s$ be an admissible block. Let d_Ω, h be as in (4.4). If the local interpolants are constructed by Chebyshev interpolation, the factorised matrix from (2.9) satisfies*

$$\begin{aligned}
\|G|_{t\times s} - U_t S_{t,s} V_s^T\|_2 \leq \frac{C_c(m) |\Omega_t|^{1/2} |\Omega_s|^{1/2} h^{d_\Omega}}{(C_{\mathrm{as2}} \operatorname{dist}(B_t, B_s))^\sigma} \left(\frac{\eta}{\eta + C_{\mathrm{as2}}}\right)^m \times \\
\times \left(\|c_0^x\| + \frac{C_{\mathrm{as2}}^{-1} \|c^x\|}{\operatorname{dist}(B_t, B_s)} \right) \times \\
\times \left(\|c_0^y\| + \frac{C_{\mathrm{as2}}^{-1} \|c^y\|}{\operatorname{dist}(B_t, B_s)} \right)
\end{aligned}$$

for a polynomial C_c that does not depend on t, s, m or h.

Proof. Combine Lemma 4.6 with Corollary 4.5 and the fact that $\Lambda_m \leq m + 1$ holds for Chebyshev interpolation (cf. [20]). ∎

4.2 HCA based on Lagrange Polynomials

In HCA(I), we replace $S_{t,s}$ by a low-rank approximation $\widetilde{S}_{t,s}$, so the total error in an admissible block $t \times s$ can be split into the sum

$$\begin{aligned}
\|G|_{t\times s} - U_t \widetilde{S}_{t,s} V_s^T\|_2 &\leq \|G|_{t\times s} - U_t S_{t,s} V_s^T\|_2 \\
&\quad + \|U_t (S_{t,s} - \widetilde{S}_{t,s}) V_s^T\|_2 \\
&\leq \|G|_{t\times s} - U_t S_{t,s} V_s^T\|_2 \\
&\quad + \|U_t\|_2 \|S_{t,s} - \widetilde{S}_{t,s}\|_2 \|V_s\|_2.
\end{aligned}$$

The first term is bounded due to Corollary 4.7, so we only have to consider the second. Here, the error $\|S_{t,s} - \widetilde{S}_{t,s}\|_2$ introduced by the ACA algorithm is scaled by the matrices $\|U_t\|_2$ and $\|V_s\|_2$ corresponding to discretised Lagrange polynomials. The ACA error can be controlled directly, so we have a complete error estimate once we can bound the latter two norms.

Lemma 4.8 (Bound of U_t) *Let $C_{\mathrm{fe}}, d_\Omega, h$ be as in (4.4). Then the matrix U_t from (2.10) satisfies*

$$\|U_t\|_2 \leq C_{\mathrm{fe}}^{1/2} \Lambda_m^3 |\Omega_t|^{1/2} h^{d_\Omega/2} \left(\|c_0^x\| + \frac{2m^2}{\operatorname{diam}(B_t)} \|c^x\| \right).$$

Proof. Let $u \in \mathbb{R}^t$ and $v \in \mathbb{R}^M$. We set

$$\hat{u} := \sum_{i \in t} u_i \varphi_i, \qquad \hat{v} := \sum_{\nu=1}^{M} v_\nu \mathcal{L}_\nu^t$$

and find

$$\begin{aligned} |\langle u, U_t v \rangle| &= \left| \int_{\Omega_t} \hat{u}(x) (D_x \hat{v})(x) \, \mathrm{d}x \right| \leq \|D_x \hat{v}\|_\infty |\Omega_t|^{1/2} \|\hat{u}\|_{L^2} \\ &\overset{(4.4)}{\leq} C_{\mathrm{fe}} |\Omega_t|^{1/2} h^{d_\Omega/2} \|D_x \hat{v}\|_\infty \|u\|_2. \end{aligned}$$

We have to bound

$$\|D_x \hat{v}\|_\infty \leq \|c_0^x\| \|\hat{v}\|_\infty + \sum_{j=1}^{3} \|c_j^x\|_\infty \|\partial_j \hat{v}\|_\infty.$$

For the first term, we use the stability (4.2) of the interpolation in order to find

$$\|\hat{v}\|_\infty \leq \Lambda_m^3 \|v\|_\infty \leq \Lambda_m^3 \|v\|_2.$$

For the remaining terms, we use Markov's inequality [10, Theorem 4.1.4] to find

$$\|\partial_i \hat{v}\|_\infty \leq \frac{2m^2}{\operatorname{diam}(B_t)} \|\hat{v}\|_\infty$$

and then use the previous estimate. We end up with the bound

$$|\langle u, U_t v \rangle| \leq C_{\mathrm{fe}}^{1/2} \Lambda_m^3 |\Omega_t|^{1/2} h^{d_\Omega/2} \left(\|c_0^x\| + \frac{2m^2}{\operatorname{diam}(B_t)} \|c^x\| \right) \|v\|_2 \|u\|_2,$$

and since this holds for all u and v, the proof is complete. ∎

Obviously this Lemma can also be used to find an upper bound for the norm $\|V_s\|_2$. The difference $\varepsilon_{\mathrm{ACA}} := \|S_{t,s} - \widetilde{S}^{t,s}\|$ is due to the ACA approximation (see Assumption 4.9) but any rank revealing scheme could be used, even a singular value decomposition of the $M \times M$ matrix $S_{t,s}$.

4.3 HCA based on Cross Approximation

In HCA(II) we replace the kernel function $g(x,y) = D_x D_y \gamma(x,y)$ by the approximant $\tilde{g} := D_x D_y \tilde{\gamma}(x,y)$ with

$$\tilde{\gamma}(x,y) := \sum_{\ell=1}^{k} \left(\sum_{q=1}^{\ell} \gamma(x, y_{j_q}) C_{\ell,q} \right) \left(\sum_{q=1}^{\ell} \gamma(x_{i_q}, y) D_{\ell,q} \right). \tag{4.5}$$

The coefficient matrices C, D above are given by the following recursion formula (see Algorithm 3):

$$\begin{aligned}
d_\iota^{(\ell)} &:= \sum_{q=1}^{\iota} \gamma(x_{i_q}, y_{j_\ell}) D_{\iota,q} \\
c_\iota^{(\ell)} &:= \sum_{q=1}^{\iota} \gamma(x_{i_\ell}, y_{j_q}) C_{\iota,q} \\
C_{\ell,q} &:= \begin{cases} |(\hat{a}_\ell)_{i_\ell}|^{-1/2} & q = \ell \\ -C_{\ell,\ell} \sum_{\iota=q}^{\ell-1} C_{\iota,q} d_\iota^{(\ell)} & q < \ell \end{cases}, \qquad (4.6) \\
D_{\ell,q} &:= \begin{cases} \operatorname{sign}((\hat{a}_\ell)_{i_\ell}) |(\hat{a}_\ell)_{i_\ell}|^{-1/2} & q = \ell \\ -D_{\ell,\ell} \sum_{\iota=q}^{\ell-1} D_{\iota,q} c_\iota^{(\ell)} & q < \ell \end{cases}. \qquad (4.7)
\end{aligned}$$

In order to estimate the approximation error $|g - \tilde{g}|$ we consider the error at first in the interpolation points for the generator function γ. Since we use the adaptive cross approximation from [1] and only the pessimistic estimate

$$\|S_{t,s} - \widehat{A}\widehat{B}^T\|_2 = \mathcal{O}(2^k \eta^{\sqrt[3]{k}})$$

has been proven, we have to check convergence and stability. The existence of a low rank cross approximation is proven in [12] which gives rise to the hope that a convergent and stable cross approximation algorithm can be found. Until then, we have to check this a posteriori.

Assumption 4.9 *We assume that the coefficients for the ACA approximation are stable in the sense*

$$C_{\ell,q}, D_{\ell,q} \ \leq \ C_\gamma,$$

where the stability constant may depend on γ, and we assume that the estimate

$$|(S_{t,s} - \widehat{A}\widehat{B}^T)_{ij}| \leq \varepsilon_{\mathrm{ACA}}$$

holds for all $i \in t, j \in s$ and a prescribed accuracy $\varepsilon_{\mathrm{ACA}}$.

If the previous Assumption 4.9 is *not fulfilled* (it is a pointwise estimate for M^2 pairs of interpolation points and can be tested in $\mathcal{O}(M^2 k)$), then we use the standard tensor interpolation in the block $t \times s$, i.e., the interpolant of γ with separation rank $k = M$ instead of $\tilde{\gamma}$. Therefore, even if ACA fails to converge for the matrix $S_{t,s}$ — which we don't expect — we still have a reliable approximation scheme.

Theorem 4.10 (Approximation Error in Interpolation Points) *Under Assumption 4.9 the function $\tilde{\gamma}$ from (4.5) with coefficients $C_{\ell,q}$ and $D_{\ell,q}$ defined in (4.6) and (4.7) satisfies*

$$|\gamma(x,y) - \tilde{\gamma}(x,y)| \leq \varepsilon_{\mathrm{ACA}} \tag{4.8}$$

for all interpolation points $x = x_1, \ldots, x_M, y = y_1, \ldots, y_M$.

Proof. We apply ACA [1, Section 2] to the function $\gamma(x,y)$. In the first step we produce the rank 1 function

$$\gamma_1(x,y) := \gamma(x, y_{j_1})\gamma(x_{i_1}, y)\gamma(x_{i_1}, y_{j_1})^{-1}.$$

The ℓth approximation $\gamma_\ell(x,y)$, $\ell = 1, \ldots, k$, will be of the form

$$\gamma_\ell(x,y) = \sum_{\iota=1}^{\ell} \left(\sum_{q=1}^{\iota} \gamma(x, y_{j_q}) C_{\iota,q} \right) \left(\sum_{q=1}^{\iota} \gamma(x_{i_q}, y) D_{\iota,q} \right), \tag{4.9}$$

which is true for the first term with

$$C_{11} = |\sigma_1|^{-1/2}, D_{11} = \mathrm{sign}(\sigma_1)|\sigma_1|^{-1/2}, \sigma_1 := \gamma(x_{i_1}, y_{j_1}) = (\hat{a}_1)_{i_1}.$$

For the induction step from ℓ to $\ell+1$ we apply the ACA step

$$\begin{aligned} \gamma_{\ell+1}(x,y) &= \gamma_\ell(x,y) + \\ &\quad + \frac{\big(\gamma(x, y_{j_{\ell+1}}) - \gamma_\ell(x, y_{j_{\ell+1}})\big)\big(\gamma(x_{i_{\ell+1}}, y) - \gamma_\ell(x_{i_{\ell+1}}, y)\big)}{\gamma(x_{i_{\ell+1}}, y_{j_{\ell+1}}) - \gamma_\ell(x_{i_{\ell+1}}, y_{j_{\ell+1}})}. \end{aligned}$$

Inserting the ansatz (4.9) and comparing the coefficients of the functions $\gamma(x, y_{j_q})$ and $\gamma(x_{i_q}, y)$ yields the desired formulae for $C_{\iota,q}$ and $D_{\iota,q}$. The approximation error is estimated in [1, Theorem 4] for all interpolation points x_i, y_i, $i = 1, \ldots, M$, and due to Assumption 4.9

$$|\gamma(x_i, y_j) - \tilde{\gamma}(x_i, y_j)| = |(S_{t,s} - AB^T)_{ij}| \leq \varepsilon_{\mathrm{ACA}}.$$

■

The previous theorem proves the convergence of $\tilde{\gamma} \to \gamma$ in the interpolation points. In order to estimate the approximation error for arbitrary points x, y, we have to relate $\tilde{\gamma}$ and γ to their respective interpolants. For γ we have required the asymptotic smoothness, while $\tilde{\gamma}$ inherits this property from γ.

Lemma 4.11 (Smoothness of $\tilde{\gamma}$) *Let γ be asymptotically smooth with constants C_{as1}, C_{as2}. The function $\tilde{\gamma}$ is asymptotically smooth with constants*

$$\begin{aligned} C_{\mathrm{as1}}(\tilde{\gamma}) &:= k^3 C_\gamma^2 C_{\mathrm{as1}}^2 (C_{\mathrm{as2}}(1+\eta)^{-1}\mathrm{dist}(B_t, B_s))^{-\sigma}, \\ C_{\mathrm{as2}}(\tilde{\gamma}) &:= (1+\eta)^{-1} C_{\mathrm{as2}}. \end{aligned}$$

Proof. Let $x \in B_t, y \in B_s$. Then the (α,β)-derivative of $\tilde{\gamma}$ is estimated by

$$\begin{aligned}|\partial_x^\alpha \partial_y^\beta \tilde{\gamma}(x,y)| &\leq \sum_{\ell=1}^{k} \left(\sum_{q=1}^{\ell} |\partial_x^\alpha \gamma(x, y_{j_q})| \, |C_{\ell,q}| \right) \times \\ &\quad \times \left(\sum_{q=1}^{\ell} |\partial_y^\beta \gamma(x_{i_q}, y)| \, |D_{\ell,q}| \right) \\ &\leq k^3 C_\gamma^2 \|\partial_x^\alpha \gamma(x,\cdot)\|_{\infty,B_s} \|\partial_y^\beta \gamma(\cdot,y)\|_{\infty,B_t}.\end{aligned}$$

Due to the admissibility condition we conclude $\|x-y\| \leq \|x-\tilde{y}\| + \operatorname{diam}(B_s) \leq \|x-\tilde{y}\| + \eta\|x-\tilde{y}\|$ and thus

$$\|\partial_x^\alpha \gamma(x,\cdot)\|_{\infty,B_s} \leq C_{\mathrm{as1}} (C_{\mathrm{as2}}(1+\eta)^{-1}\|x-y\|)^{-|\alpha|-\sigma} \alpha!,$$

analogously for the ∂_y^β derivative. Both together yield

$$|\partial_x^\alpha \partial_y^\beta \tilde{\gamma}(x,y)| \leq C_{\mathrm{as1}}^2 (C_{\mathrm{as2}}(1+\eta)^{-1}\|x-y\|)^{-|\alpha|-|\beta|-2\sigma} \alpha!\beta!$$

■

Corollary 4.12 (Approximation Error) *Under Assumption 4.9 the function $\tilde{\gamma}$ with coefficients $C_{\ell,q}, D_{\ell,q}$ defined in (4.6) and (4.7) satisfies*

$$\|\gamma - \tilde{\gamma}\|_{\infty,B_t\times B_s} \leq \Lambda_m^{2d} \, \varepsilon_{\mathrm{ACA}} + 2\varepsilon_{\mathrm{int}}, \tag{4.10}$$

where $\varepsilon_{\mathrm{int}}$ is the interpolation error that is estimated separately in Theorem 4.4.

Proof. We apply the tensor interpolation $I_m^{B_t\times B_s}$ to the asymptotically smooth functions γ and $\tilde{\gamma}$ and define $\gamma_{\mathrm{int}} := I_m^{B_t\times B_s}[\gamma], \tilde{\gamma}_{\mathrm{int}} := I_m^{B_t\times B_s}[\tilde{\gamma}]$. Both interpolants fulfil the estimate

$$\begin{aligned}\|\gamma - \gamma_{\mathrm{int}}\|_{\infty,B_t\times B_s} &\leq \varepsilon_{\mathrm{int}}, \\ \|\tilde{\gamma} - \tilde{\gamma}_{\mathrm{int}}\|_{\infty,B_t\times B_s} &\leq \varepsilon_{\mathrm{int}}\end{aligned}$$

with an accuracy $\varepsilon_{\mathrm{int}}$ estimated in Theorem 4.4. Since both interpolants use the same interpolation points and the difference between γ and $\tilde{\gamma}$ in the interpolation points is bounded in Theorem 4.10 by $\varepsilon_{\mathrm{ACA}}$, the stability of the interpolation scheme yields

$$\|\gamma_{\mathrm{int}} - \tilde{\gamma}_{\mathrm{int}}\|_{\infty,B_t\times B_s} \leq \Lambda_m^{2d} \, \varepsilon_{\mathrm{ACA}}.$$

■

Theorem 4.13 (Separable Approximation) *Under Assumption 4.9 the function*

$$\tilde{g}(x,y) := \sum_{\ell=1}^{k} \left(\sum_{q=1}^{\ell} g(x, y_{j_q}) C_{\ell,q} \right) \left(\sum_{q=1}^{\ell} g(x_{i_q}, y) D_{\ell,q} \right). \tag{4.11}$$

with coefficients $C_{\ell,q}, D_{\ell,q}$ defined in (4.6) and (4.7) satisfies

$$\begin{aligned}\|g-\tilde{g}\|_{\infty,B_t\times B_s} &\leq \left(\|c_0^x\|+\frac{2m^2\|c^x\|}{\operatorname{diam}(B_t)}\right)\left(\|c_0^y\|+\frac{2m^2\|c^y\|}{\operatorname{diam}(B_s)}\right)\Lambda_m^{2d}\varepsilon_{\mathrm{ACA}}\\ &\quad +2\varepsilon_{\mathrm{int}},\end{aligned}$$

where $\varepsilon_{\mathrm{int}}$ is the interpolation error of the derivatives estimated separately in Corollary 4.5.

Proof. By definition we have $g = D_xD_y\gamma$ and $\tilde{g} = D_xD_y\tilde{\gamma}$. We apply the tensor interpolation $I_m^{B_t\times B_s}$ to the asymptotically smooth functions γ and $\tilde{\gamma}$ and define $\gamma_{\mathrm{int}} := I_m^{B_t\times B_s}[\gamma], \tilde{\gamma}_{\mathrm{int}} := I_m^{B_t\times B_s}[\tilde{\gamma}]$. Both interpolants fulfil the estimate

$$\begin{aligned}\|D_xD_y\gamma - D_xD_y\gamma_{\mathrm{int}}\|_{\infty,B_t\times B_s} &\leq \varepsilon_{\mathrm{int}},\\ \|D_xD_y\tilde{\gamma} - D_xD_y\tilde{\gamma}_{\mathrm{int}}\|_{\infty,B_t\times B_s} &\leq \varepsilon_{\mathrm{int}}\end{aligned}$$

with an accuracy $\varepsilon_{\mathrm{int}}$ estimated in Corollary 4.5. It remains to estimate the difference between the interpolants $g_{\mathrm{int}} := D_xD_y\gamma_{\mathrm{int}}$ and $\tilde{g}_{\mathrm{int}} := D_xD_y\tilde{\gamma}_{\mathrm{int}}$:

$$\begin{aligned}\|g_{\mathrm{int}}-\tilde{g}_{\mathrm{int}}\|_{\infty,B_t\times B_s} &\overset{(2.3)}{\leq} \|c_0^x\|\|c_0^y\|\|\gamma_{\mathrm{int}}-\tilde{\gamma}_{\mathrm{int}}\|_{\infty,B_t\times B_s}\\ &\quad +\|c_0^x\|\|\langle c^y,\nabla_y(\gamma_{\mathrm{int}}-\tilde{\gamma}_{\mathrm{int}})\rangle\|_{\infty,B_t\times B_s}\\ &\quad +\|c_0^y\|\|\langle c^x,\nabla_x(\gamma_{\mathrm{int}}-\tilde{\gamma}_{\mathrm{int}})\rangle\|_{\infty,B_t\times B_s}\\ &\quad +\|\langle c^x,\nabla_x\langle c^y,\nabla_y(\gamma_{\mathrm{int}}-\tilde{\gamma}_{\mathrm{int}})\rangle\rangle\|_{\infty,B_t\times B_s}\\ &\overset{\text{Markov}}{\leq} \|c_0^x\|\|c_0^y\|\Lambda_m^{2d}\,\varepsilon_{\mathrm{ACA}}\\ &\quad +\|c_0^x\|\|c^y\|\frac{2m^2}{\operatorname{diam}(B_s)}\Lambda_m^{2d}\varepsilon_{\mathrm{ACA}}\\ &\quad +\|c_0^y\|\|c^x\|\frac{2m^2}{\operatorname{diam}(B_t)}\Lambda_m^{2d}\varepsilon_{\mathrm{ACA}}\\ &\quad +\|c^x\|\|c^y\|\frac{4m^4}{\operatorname{diam}(B_t)\operatorname{diam}(B_s)}\Lambda_m^{2d}\varepsilon_{\mathrm{ACA}}.\end{aligned}$$

■

Corollary 4.14 (Cross Approximation) *Let $t\times s$ be an admissible block. Let C_{fe}, d_Ω, h be as in (4.4). If the degenerate kernel approximation is based on the cross approximation (4.11), the factorised matrix AB^T from Algorithm 3 satisfies*

$$\begin{aligned}\|G|_{t\times s}-AB^T\|_2 &\leq C_{\mathrm{fe}}|\Omega_t|^{1/2}|\Omega_s|^{1/2}h^{d_\Omega}\Bigg(2\varepsilon_{\mathrm{int}}+\Lambda_m^{2d}\varepsilon_{\mathrm{ACA}}\times\\ &\quad\times\left(\|c_0^x\|+\frac{2m^2\|c^x\|}{\operatorname{diam}(B_t)}\right)\left(\|c_0^y\|+\frac{2m^2\|c^y\|}{\operatorname{diam}(B_s)}\right)\Bigg).\end{aligned}$$

Proof. The estimate from Lemma 4.6 applies in the same way and with the same estimates for $\tilde{g}$ constructed by cross approximation and AB^T instead of $U_tS_{t,s}V_s^T$. The error due to the degenerate kernel approximation is bounded in Theorem 4.13. ■

5 Numerical Results

We will now consider a more complicated (and more realistic) geometry, namely the crank shaft from the NETGEN package of Joachim Schöberl (see Figure 4). For all numerical tests reported in the tables of this section, "Time" gives the time in seconds for constructing the approximation (including the automatic coarsening from [14]), "Strg." gives the storage required for the resulting $\mathcal{H}$-matrix in kilobytes per degree of freedom and "Rel. Err." gives the relative error $\|I - \widetilde{G}^{-1}G\|_2$ approximated using a power iteration and a sufficiently accurate representation of the stiffness matrix G.

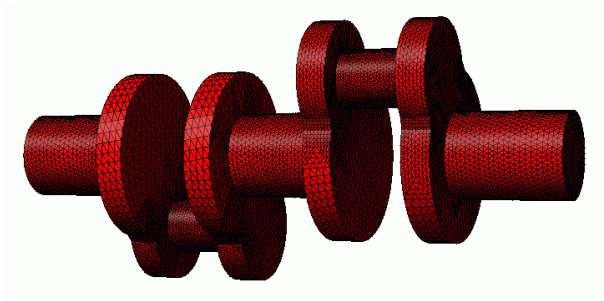


Figure 4: The crank shaft geometry from NETGEN.

In order to compare our new approximation scheme with a working heuristic, we have to modify Algorithm 1 (ACA). This is done by intercepting the situation from Example 2.2 as well as the situations from [14, Example 2.3] depicted in Figure 5. We do this by

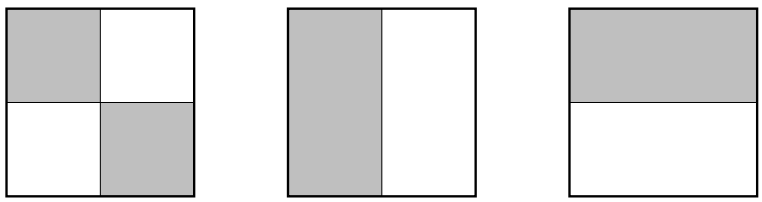


Figure 5: Three situations where ACA fails to converge (the shaded regions are the non-zero parts of the matrix)

inspecting an arbitrary row i_{ref} of the matrix block $G|_{t\times s}$ to be assembled. A column index j_{ref} is chosen among the minimisers (not maximisers !) of $|G_{i_{\text{ref}},j}|$. The row i_{ref} and column j_{ref} serve as a pivoting guide in ACA+ [14]. For the sake of completeness the ACA+ algorithm is contained in Algorithm 4. It should be noted that the above mentioned modification of ACA remedies the three situations from Figure 5 but it does not prove convergence, not even the pessimistic $\mathcal{O}(2^k\eta^{\sqrt[3]{k}})$ estimate.

Algorithm 4 ACA+ with partial pivoting

procedure ACA+(X, **var** A, B)
Choose an initial reference row i_{ref} and compute the entries $b_j^{\mathrm{ref}} := X_{i_{\mathrm{ref}},j}$ of $b^{\mathrm{ref}} \in \mathbb{R}^m$
Determine an index j_{ref} that minimises $|b_{j_{\mathrm{ref}}}^{\mathrm{ref}}|$
Compute the entries $a_i^{\mathrm{ref}} := X_{i,j_{\mathrm{ref}}}$ of $a^{\mathrm{ref}} \in \mathbb{R}^n$
$k := 1$, $I_{\mathrm{ref}} := \{1, \ldots, n\}$, $J_{\mathrm{ref}} := \{1, \ldots, m\}$
if $\min_i |a_i^{\mathrm{ref}}| < 10^{-8} \max_i |a_i^{\mathrm{ref}}|$ and $\min_j |b_j^{\mathrm{ref}}| < 10^{-8} \max_j |b_j^{\mathrm{ref}}|$ **then**
$I_{\mathrm{ref}} := I_{\mathrm{ref}} \setminus \{i \mid |a_i^{\mathrm{ref}}| > 10^8 \min_i |a_i^{\mathrm{ref}}|\}$ and $J_{\mathrm{ref}} := J_{\mathrm{ref}} \setminus \{j \mid |b_j^{\mathrm{ref}}| > 10^8 \min_j |b_j^{\mathrm{ref}}|\}$
end if
repeat
if $\max_i |a_i^{\mathrm{ref}}| > \max_j |b_j^{\mathrm{ref}}|$ **then**
Determine an index i_k^* that maximises $|a_{i_k^*}^{\mathrm{ref}}|$ and compute $b_k \in \mathbb{R}^m$ by

$$(b_k)_j := X_{i_k^*,j} - \sum_{\mu=1}^{k-1} (a_\mu)_{i_k^*} (b_\mu)_j.$$

Determine an index j_k^* that maximises $|(b_k)_{j_k^*}|$ and compute $a_k \in \mathbb{R}^n$ by

$$(a_k)_i := \left(X_{i,j_k^*} - \sum_{\mu=1}^{k-1} (a_\mu)_i (b_\mu)_{j_k^*} \right) / (b_k)_{j_k^*}$$

else
Determine an index j_k^* that maximises $|b_{j_k^*}^{\mathrm{ref}}|$ and compute $a_k \in \mathbb{R}^n$ by

$$(a_k)_i := X_{i,j_k^*} - \sum_{\mu=1}^{k-1} (a_\mu)_i (b_\mu)_{j_k^*}.$$

Determine an index i_k^* that maximises $|(a_k)_{i_k^*}|$ and compute $b_k \in \mathbb{R}^m$ by

$$(b_k)_j := \left(X_{i_k^*,j} - \sum_{\mu=1}^{k-1} (a_\mu)_{i_k^*} (b_\mu)_j \right) / (a_k)_{i_k^*}$$

end if
if $i_k^* = i_{\mathrm{ref}}$ **then**
Choose $i_{\mathrm{ref}} \in I_{\mathrm{ref}}$ and set $I_{\mathrm{ref}} := I_{\mathrm{ref}} \setminus \{i_{\mathrm{ref}}\}$
end if
if $j_k^* = j_{\mathrm{ref}}$ **then**
Choose $j_{\mathrm{ref}} \in J_{\mathrm{ref}}$ and set $J_{\mathrm{ref}} := J_{\mathrm{ref}} \setminus \{j_{\mathrm{ref}}\}$
end if
Update the reference entries $a_i^{\mathrm{ref}} := a_i^{\mathrm{ref}} - (a_k)_i (b_k)_{j_k^*}$ and $b_j^{\mathrm{ref}} := b_j^{\mathrm{ref}} - (a_k)_{i_k^*} (b_k)_j$
$k := k + 1$
until $\|a_k\|_2 \|b_k\|_2 \leq \epsilon \|a_1\|_2 \|b_1\|_2$

ACA			HCA(II)		
Time	Strg.	Rel. Err.	Time	Strg.	Rel. Err.
333	4.9	1.9×10^{-1}	362	5.0	2.0×10^{-1}
449	8.9	1.8×10^{-2}	537	8.9	1.7×10^{-2}
837	14.5	3.4×10^{-3}	796	13.2	1.6×10^{-3}
2306	42.0	1.8×10^{-4}	1151	18.7	1.5×10^{-4}

Table 4: Comparison of ACA and HCA(II) on the Crank Shaft

5.1 Dependency on the Accuracy of the Quadrature

In our first numerical test we investigate the dependency of the compression by ACA and HCA(II) on the accuracy of the quadrature used in the farfield. Typically, one uses a given quadrature formula and chooses the accuracy for the compression so that it is below the quadrature error. In the farfield one can even lower the order of the quadrature, because the integrand is smooth compared to the diameter of the elements. In the numerical test we try to approximate the stiffness matrix G (assembled with a fixed quadrature) with $n = 25744$ degrees of freedom by ACA and HCA(II) with increasing accuracy

The results in Table 4 indicate that ACA is deteriorated by the quadrature error and produces approximations with much higher storage requirements. Both methods yield an approximation of approximately the same quality in almost the same time — except for the higher accuracy where ACA is deteriorated. We conclude that for ACA one should choose a sufficiently accurate quadrature formula and omit the lowering of the quadrature order in the farfield.

The situation is different for HCA. Since we separate the kernel in the bounding box before applying the quadrature, the blockwise rank produced by HCA is independent of the quadrature order, the basis functions or the element size.

5.2 Dependency on the Meshsize

According to Lemma 2.12, we expect the storage requirements as well as the time for the setup of an $\mathcal{H}$-matrix approximation with fixed accuracy (rank) to be $\mathcal{O}(n\log(n))$, i.e., an increase per degree of freedom as the mesh is regularly refined. This behaviour can be observed in Table 5 where we approximate the stiffness matrix on three different levels with $n = 25744, 102976, 411904$ degrees of freedom. The order of the quadrature is increased from the first to the second row and from the third to the fourth row.

References

[1] Mario Bebendorf. Approximation of boundary element matrices. *Numer. Math.*, 86(4):565–589, 2000.

$n = 25744$			$n = 102976$			$n = 411904$		
t	Strg.	Rel. Err.	t	Strg.	Rel. Err.	t	Strg.	Rel. Err.
.27	4.9	1.9×10^{-1}	1.06	7.0	1.6×10^{-1}	4.45	8.4	1.6×10^{-1}
.56	8.9	1.5×10^{-2}	2.30	12.2	1.3×10^{-2}	9.72	14.7	1.4×10^{-2}
.83	13.2	1.1×10^{-3}	3.43	18.2	1.2×10^{-3}	14.77	21.9	1.7×10^{-3}
1.73	18.6	7.4×10^{-5}	7.09	25.4	6.6×10^{-5}	30.04	30.6	7.0×10^{-5}

Table 5: HCA(II) on the Crank Shaft with increasing number of degrees of freedom, the elapsed time t is measured in 1000 seconds

[2] Mario Bebendorf and Sergej Rjasanov. Adaptive Low-Rank Approximation of Collocation Matrices. *Computing*, 70(1):1–24, 2003.

[3] Steffen Börm. Approximation of integral operators by $\mathcal{H}^2$-matrices with adaptive bases. Technical Report 18, Max Planck Institute for Mathematics in the Sciences, 2004. To appear in Computing.

[4] Steffen Börm and Lars Grasedyck. HLib – a library for $\mathcal{H}$- and $\mathcal{H}^2$-matrices, 1999. Available at `http://www.hlib.org/`.

[5] Steffen Börm and Lars Grasedyck. Low-rank approximation of integral operators by interpolation. *Computing*, 72:325–332, 2004.

[6] Steffen Börm, Lars Grasedyck, and Wolfgang Hackbusch. Introduction to hierarchical matrices with applications. *Engineering Analysis with Boundary Elements*, 27:405–422, 2003.

[7] Steffen Börm and Wolfgang Hackbusch. Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices. *Computing*, 69:1–35, 2002.

[8] Steffen Börm and Wolfgang Hackbusch. $\mathcal{H}^2$-matrix approximation of integral operators by interpolation. *Applied Numerical Mathematics*, 43:129–143, 2002.

[9] Wolfgang Dahmen and Reinhold Schneider. Wavelets on manifolds I: Construction and domain decomposition. *SIAM Journal of Mathematical Analysis*, 31:184–230, 1999.

[10] Ronald A. DeVore and George G. Lorentz. *Constructive Approximation*. Springer-Verlag, 1993.

[11] J. M. Ford and E. E. Tyrtyshnikov. Combining kronecker product approximation with discrete wavelet transforms to solve dense, function-related linear systems. *SIAM J. Sci. Comput.*, 25(3):961–981, 2003.

[12] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin. A theory of pseudoskeleton approximations. *Lin. Alg. Appl.*, 261:1–22, 1997.

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

[14] Lars Grasedyck. Adaptive recompression of $\mathcal{H}$-matrices for BEM. Technical Report 17, Max Planck Institute for Mathematics in the Sciences, 2004.

[15] Lars Grasedyck and Wolfgang Hackbusch. Construction and arithmetics of $\mathcal{H}$-matrices. *Computing*, 70(4):295–334, 2003.

[16] Leslie Greengard and Vladimir Rokhlin. A new version of the fast multipole method for the Laplace in three dimensions. In *Acta Numerica 1997*, pages 229–269. Cambridge University Press, 1997.

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

[18] Wolfgang Hackbusch and Boris Khoromskij. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part II: Application to multi-dimensional problems. *Computing*, 64:21–47, 2000.

[19] Wolfgang Hackbusch and Zenon Paul Nowak. On the fast matrix multiplication in the boundary element method by panel clustering. *Numerische Mathematik*, 54:463–491, 1989.

[20] T.J. Rivlin. *The Chebyshev Polynomials*. Wiley-Interscience, New York, 1984.

[21] Vladimir Rokhlin. Rapid solution of integral equations of classical potential theory. *Journal of Computational Physics*, 60:187–207, 1985.

[22] Stefan Sauter. Variable order panel clustering (extended version). Technical Report 52, Max-Planck-Institut für Mathematik, Leipzig, Germany, 1999.

[23] Eugene Tyrtyshnikov. Incomplete cross approximation in the mosaic-skeleton method. *Computing*, 64:367–380, 2000.