Max-Planck-Institut
für Mathematik
in den Naturwissenschaften
Leipzig

Adaptive Geometrically Balanced Clustering of $\mathcal{H}$-Matrices

by

Lars Grasedyck, Wolfgang Hackbusch, and Sabine Le Borne

Preprint no.: 19 2004

Adaptive Geometrically Balanced Clustering of $\mathcal{H}$-Matrices

L. Grasedyck and W. Hackbusch, Leipzig
S. Le Borne, Cookeville

Abstract

In [8], a class of (data-sparse) hierarchical ($\mathcal{H}$-) matrices is introduced that can be used to efficiently assemble and store stiffness matrices arising in boundary element applications. In this paper, we develop and analyse modifications in the construction of an $\mathcal{H}$-matrix that will allow an efficient application to problems involving adaptive mesh refinement. In particular, we present a new clustering algorithm such that, when an $\mathcal{H}$-matrix has to be updated due to some adaptive grid refinement, the majority of the previously assembled matrix entries can be kept whereas only a few new entries resulting from the refinement have to be computed. We provide an efficient implementation of the necessary updates and prove for the resulting $\mathcal{H}$-matrix that the storage requirements as well as the complexity of the matrix-vector multiplication are almost linear, i.e., $\mathcal{O}(n \log(n))$.

AMS Subject Classification: *65F05, 65F30, 65N38, 65N50*
Key words: *Hierarchical matrices, data-sparse approximation, adaptive mesh refinement, boundary elements*

1 Introduction

In the following, we want to treat boundary element matrices in an efficient way using the format of hierarchical matrices (as explained below). The modern approach to discretisation methods is an adaptive one. Given an intermediate boundary element triangulation on the surface, the problem is solved approximately and an error estimator or indicator is used to predict where to refine the triangulation (cf. [4]). Often, the refinement is only local (e.g., along edges of the surface). As a result, the boundary element matrices before and after the refinement coincide in most of the matrix entries which therefore need not be recomputed. Since the format of hierarchical matrices uses a particularly defined block partitioning, the question arises whether this partitioning has to be redefined completely or whether it can be repaired locally. It is the purpose of this paper to show that, indeed, the internal format requires only local changes, so that also the work for the overhead is proportional to the percentage of newly introduced degrees of freedom.

The construction of an $\mathcal{H}$-matrix $M \in \mathbb{R}^{I \times I}$ is based on a hierarchical block partitioning of the product index set $I \times I$ which itself is based on a hierarchical partitioning of the index set I. These hierarchical partitionings are organised in so-called block cluster trees. In this paper, we introduce a geometrically balanced (regular) clustering algorithm to build a cluster tree and block cluster tree for which we can prove that the resulting $\mathcal{H}$-matrix has a storage and matrix-vector multiplication complexity of $\mathcal{O}(n \log n)$. We describe how the addition or removal of a relatively small number of indices (due to adaptive mesh refinement) can be efficiently realised for this type of (block) cluster tree, and, most importantly, we show that in this case most matrix entries of the original matrix can be kept and only few entries have to be recomputed for the updated matrix.

The rest of the paper is organised as follows: in Section 2, we introduce the model integral equation and the general concept of $\mathcal{H}$-matrices. In Section 3, we provide the new construction of a geometrically balanced cluster tree and derive estimates for the complexities of storage and matrix-vector multiplication for the resulting $\mathcal{H}$-matrices. In Section 4, we then illustrate how an $\mathcal{H}$-matrix is updated after an adaptive mesh refinement. In Section 5, we provide numerical results for our proposed update scheme.

2 Preliminaries

2.1 Model Problem: Integral Equation

We consider a Fredholm integral operator of the form

$$\mathcal{G}[u](x) = \int_\Omega g(x,y)u(y)\, \mathrm{d}y \tag{1}$$

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

$$g : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}.$$

$\mathcal{H}$-matrices are based on the fact that, at least for typical kernel functions $g(\cdot,\cdot)$, singularities only occur at the diagonal and the function is smooth everywhere else. In order to describe this property precisely, we introduce a special notion of *asymptotic smoothness*: the kernel function $g(\cdot,\cdot)$ is called *asymptotically smooth*, if there exist constants C_{as1} and C_{as2} and a singularity degree $\sigma \geq 0$ such that for all directions $z \in \{x_j, y_j\}$ there holds the inequality

$$|\partial_z^\nu g(x,y)| \leq C_{\mathrm{as1}}(C_{\mathrm{as2}}\|x-y\|)^{-\nu-\sigma}\nu!. \tag{2}$$

This kind of operator occurs, e.g., in the integral equation formulation of the Poisson problem in $\mathbb{R}^3$, where g is the singularity function $g(x,y) = \frac{1}{4\pi}\|x-y\|^{-1}$. A standard Galerkin discretisation of $\mathcal{G}$ for a basis $(\varphi_i)_{i\in I}$, $I = \{1,\dots,n\}$, yields a matrix G with entries

$$G_{ij} := \int_\Omega \int_\Omega \varphi_i(x) g(x,y) \varphi_j(y)\, \mathrm{d}x\, \mathrm{d}y. \tag{3}$$

Since, in general, the support of the kernel g is not local, G is a dense matrix.

The algorithmic complexity for computing and storing a dense matrix is quadratic in the number of degrees of freedom, therefore different approaches have been introduced to handle this kind of matrices: for translation-invariant kernel functions and simple geometries, the matrix G has Toeplitz structure, which can be exploited by algorithms based on the fast Fourier transformation.

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 [3].

Our approach is a refined combination of the panel clustering method [10] and hierarchical matrices [2, 8, 9], which are based on the idea of replacing the kernel function locally by degenerate approximations. On the discrete level this corresponds to the approximation of certain matrix blocks by low rank matrices. This method is especially well suited to treat non-quasiuniform grids that arise due to an adaptive refinement.

2.2 Low Rank Approximation

The success of low rank approximations of certain matrix blocks depends on the smoothness properties of the given kernel function (cf. (2)) together with a particular clustering of the index set as described next. We will use the following notation: let $t \times s \subseteq I \times I$ be a sub-block of the product index set $I \times I$. We define the corresponding domains Ω_t, Ω_s as the unions of the supports of the respective basis functions φ_i, i.e.,

$$\Omega_t := \cup_{i\in t}\mathrm{supp}(\varphi_i), \quad \Omega_s := \cup_{i\in s}\mathrm{supp}(\varphi_i). \tag{4}$$

Let B_t, B_s be axially parallel boxes containing Ω_t, Ω_s, resp. (see Figure 1). We assume that $\mathrm{dist}(B_t, B_s) > 0$ holds, which implies that $g|_{B_t\times B_s}$ is smooth. In order to ensure a uniform smoothness independent of the sets B_t and B_s, we impose the *admissibility condition*

$$\min\{\mathrm{diam}(B_t), \mathrm{diam}(B_s)\} \leq \eta\, \mathrm{dist}(B_t, B_s) \tag{5}$$

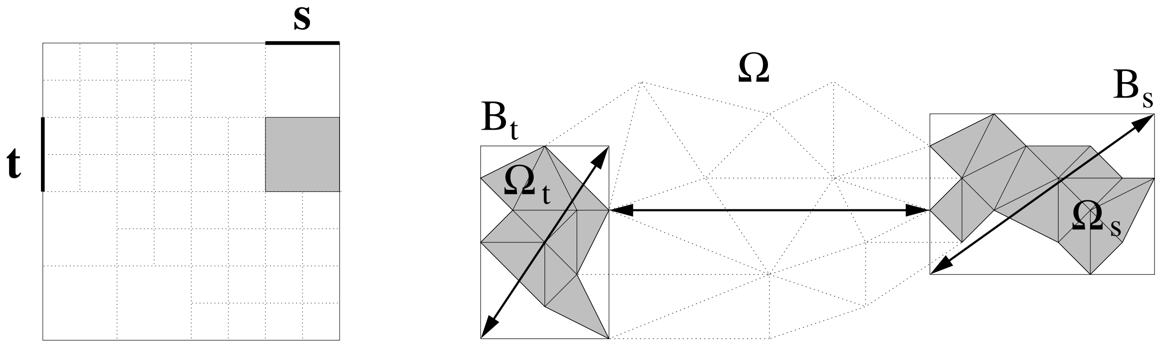


Figure 1: Each block $t \times s \subseteq I \times I$ corresponds to a subset $\Omega_t \times \Omega_s$ of $\Omega \subseteq \mathbb{R}^d$.

for some parameter $\eta > 0$. On $B_t \times B_s$, we then replace the kernel $g(x, y)$ by a tensor Lagrange interpolation polynomial with respect to the x-variable (analogously for the y-variable)

$$\tilde{g}(x, y) := \sum_{\nu \in \{0,\dots,m\}^d} g(x_\nu, y) L_\nu(x). \tag{6}$$

The approximation error (using the stable Chebyshev interpolation scheme) is estimated in

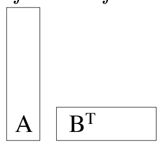
Lemma 1 (Lemma 4.1 in [1]) *Let B_t, B_s be admissible with respect to (5) and let the kernel g be asymptotically smooth as defined in (2). Then there exists a constant C_g such that for each $m \in \mathbb{N}$ the matrix $M := G|_{t\times s}$ can be approximated by a matrix $\tilde{M}$ of rank at most $(m+1)^d$ such that the error in the Frobenius norm is bounded by*

$$\|M - \tilde{M}\|_F \le C_g \sqrt{\#t\,\#s}(m+1)^{d+1} \operatorname{dist}(B_t, B_s)^{-\sigma} \max_{i \in t \cup s} \|\varphi_i\|^2_{L^1} (1 + 2C_{\mathrm{as}2}\eta^{-1})^{-(m+1)}.$$

The exponential convergence with respect to the parameter m in Lemma 1 allows us to approximate each admissible block by a matrix of very low rank (as compared to the sizes $\#t, \#s$). The key tool for the data sparse representation of this type of matrices is the $R(k)$-matrix format defined in the next section.

2.3 R(k)-matrices

Definition 2 (R(k)-matrix representation) *Let $k, n, m \in \mathbb{N}_0$. Let $M \in \mathbb{R}^{n\times m}$ be a matrix of at most rank k. A representation of M in factorised form*



$$M = AB^T, \qquad A \in \mathbb{R}^{n\times k}, B \in \mathbb{R}^{m\times k}, \tag{7}$$

with A and B stored in full matrix representation, is called an R(k)-matrix representation of M, or, in short, we call M an R(k)-matrix.

Throughout this paper the storage is measured by the number of floating point numbers to be stored, while the cost of an operation is given by the number of elementary operations $+, -, \cdot, /$.

Remark 3 (Storage and matrix-vector product) *The storage requirements $N_{F,St}(n, m)$ for a matrix $M \in \mathbb{R}^{n\times m}$ in full matrix representation is $N_{F,St}(n, m) = nm$. The storage requirements $N_{R,St}(n, m, k)$ for an $n \times m$ R(k)-matrix M is $N_{R,St}(n, m, k) = k(n + m)$. The complexities $N_{F\cdot v}(n, m)$ and $N_{R\cdot v}(n, m, k)$ for the computation of the matrix-vector product of M in full matrix and R(k)-matrix representation are*

$$N_{F\cdot v}(n, m) = 2nm - n, \quad N_{R\cdot v}(n, m, k) = 2k(n + m) - n - k.$$

If the rank k is small compared to the matrix size given by n and m, we therefore obtain considerable savings in the storage and work complexities of an $\mathrm{R}(k)$-matrix compared to a full matrix.

2.4 $\mathcal{H}$-matrices

An $\mathcal{H}$-matrix approximation to a given (full) matrix is obtained by replacing certain blocks of the matrix by $\mathrm{R}(k)$-matrices. The formal Definition 8 of an $\mathcal{H}$-matrix depends on appropriate partitionings of the index set and also the product index set which are organised in a so-called (block) cluster tree as defined next. Instead of fixed partitionings, these trees will provide hierarchies of partitionings which is the reason why the resulting matrices are called $\mathcal{H}$ierarchical matrices.

Definition 4 (Cluster tree) *Let I be a finite index set and let $T_I = (V, E)$ be a tree with vertex set V and edge set E. For a vertex $v \in V$ we define the set of sons of v as $S(v) := \{w \in V \mid (v, w) \in E\}$. The tree T_I is called a cluster tree of I if its vertices consist of subsets of I and satisfy the following conditions:*

1. $I \in V$ is the root of T_I and $v \subset I$, $v \neq \emptyset$, for all $v \in V$.

2. For all $v \in V$ there either holds $S(v) = \emptyset$ or $v = \dot{\bigcup}_{w \in S(v)} w$.

In the following we identify V and T_I, i.e., we write $v \in T_I$ instead of $v \in V$. The nodes $v \in V$ are called clusters.

For regular grids one can construct the cluster tree T_I in a cardinality balanced way, i.e., an index cluster is divided into a certain number of sons of approximately the same size with respect to the number of indices (see [8, 9]). For locally refined grids the results from [7] indicate that the cardinality balanced clustering is *not* optimal. Instead, the geometrically balanced approach looks advantageous, which will be used in an adapted version in Section 3.

Definition 5 (Leaf, father, level, depth) *Let T_I be a cluster tree. The* set of leaves *of the tree T_I is $\mathcal{L}(T_I) = \{v \in T_I \mid S(v) = \emptyset\}$. The uniquely determined predecessor (*father*) of a non-root vertex $v \in T_I$ is denoted by $\mathcal{F}(v)$. The* levels *of the tree T_I are defined by*

$$T_I^{(0)} := \{I\}, \quad T_I^{(\ell)} := \{v \in T_I \mid \mathcal{F}(v) \in T_I^{(\ell-1)}\} \qquad \text{for } \ell \in \mathbb{N},$$

and we write $\mathrm{level}(v) = \ell$ *if $v \in T_I^{(\ell)}$. The depth of T is defined as* $\mathrm{depth}(T) := \max\{\ell \in \mathbb{N}_0 \mid T_I^{(\ell)} \neq \emptyset\}$. *The leaves on level $\ell = 0, \dots, \mathrm{depth}(T)$ are*

$$\mathcal{L}(T_I, \ell) := \mathcal{L}(T_I) \cap T_I^{(\ell)}.$$

In the above definition of a cluster tree the nodes are labeled by subsets of the index set I. This implies that a node v cannot have exactly one son w since part 2 of the definition would yield $v = w$. If, however, such a case is desired, we can denote a vertex of a cluster tree by a tuple (v, ℓ) where $v \subset I$ and ℓ is the level of the node.

Remark 6 *For any cluster tree T_I and $\ell \in \{0, \dots, \mathrm{depth}(T)\}$ there holds*

$$I = \dot{\bigcup}\{v \mid v \in T_I^{(\ell)} \cup \mathcal{L}(T_I, \ell-1) \cup \dots \cup \mathcal{L}(T_I, 0)\}, \quad \textit{in particular} \quad I = \dot{\bigcup}\{v \mid v \in \mathcal{L}(T_I)\}.$$

A hierarchy of block partitionings of the product index set $I \times I$ is based upon a cluster tree T_I and is organised in a block cluster tree:

Definition 7 (Block cluster tree) *Let T_I be a cluster tree of the index set I. A cluster tree $T_{I \times I}$ is called a block cluster tree (based upon T_I) if for all $v \in T_{I \times I}^{(l)}$ there exist $t, s \in T_I^{(l)}$ such that $v = t \times s$. The nodes $v \in T_{I \times I}$ are called block clusters.*

A canonical construction of a block cluster tree $T_{I\times I}$ from a given cluster tree T_I will follow in Construction 12.

Definition 8 ($\mathcal{H}$-matrix) *Let $k, n_{\min} \in \mathbb{N}_0$. The set of $\mathcal{H}$-matrices induced by a block cluster tree $T_{I\times I}$ with blockwise rank k and minimum block size $n_{\min}$ is defined by*

$$\mathcal{H}(T_{I\times I}, k) := \{M \in \mathbb{R}^{I\times I} \mid \forall t \times s \in \mathcal{L}(T) : \operatorname{rank}(M|_{t\times s}) \leq k \textit{ or } \min\{\#t, \#s\} \leq n_{\min}\}.$$

A matrix $M \in \mathcal{H}(T_{I\times I}, k)$ is said to be given in $\mathcal{H}$-matrix representation if the blocks $M|_{t\times s}$ with $\operatorname{rank}(M|_{t\times s}) \leq k$ are in $R(k)$-matrix representation (and the remaining blocks with $\min\{\#t, \#s\} \leq n_{\min}$ are stored as full matrices). The set of indices $(i, j) \in I \times I$ that belong to $R(k)$-matrix blocks is called the farfield *while the complement is called the* nearfield.

Both the accuracy and (storage) complexity of an $\mathcal{H}$-matrix approximation to a given matrix depend on the construction of an appropriate cluster tree, i.e., a hierarchy of index set partitionings. Lemma 1 provides the approximation error for blocks that satisfy the admissibility condition, i.e., for blocks that have a large distance compared to their diameters, whereas Remark 3 provides the storage requirements for full as well as R(k)-matrices. Therefore, the intuitive objective in the construction of a cluster tree is to partition the index set into clusters of vertices that are geometrically close to each other. As a result, relatively large blocks become admissible and we obtain an accurate $\mathcal{H}$-matrix approximation that is inexpensive to store. In the following section we provide a new algorithm for the construction of a cluster tree. Results on storage and work complexities for the resulting $\mathcal{H}$-matrices will follow in Section 3.4.

3 Geometrically Balanced Clustering and $\mathcal{H}$-matrix Properties

3.1 Construction of the Cluster Tree T_I

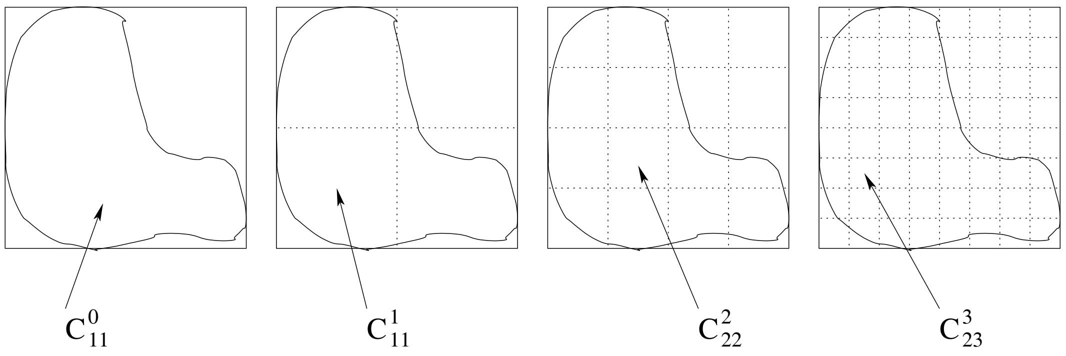
In previous papers (e.g., [9], [2], [7]), a cluster tree has been constructed in a cardinality balanced way. In Example 17, we show why this construction is not suitable in the case of adaptively refined grids. In particular, the addition of only a few new indices might lead to a completely new and different partition and therefore require the costly (re-) computation of all matrix entries in the farfield (and not only those corresponding to the new indices).

Here, we will provide the construction of a geometrically balanced (regular) cluster tree, which is a modification of the construction introduced in [7] in order to estimate the complexity of the $\mathcal{H}$-matrix arithmetic. In subsequent sections we will illustrate why these trees are well-suited for the case of adaptively refined grids: individual indices can be efficiently inserted or removed from a given tree, and as a result only very few matrix entries have to be computed for the update whereas most of the previously assembled entries can be kept.

In the following construction, m_i denotes the Chebyshev centre of the support of the basis function φ_i associated with the index i (the Chebyshev centre of a set is the centre of the smallest ball containing the set).

Construction 9 (Geometrically balanced clustering) *Without loss of generality we assume that the domain Ω is contained in the cube $[0, h_{\max})^d$. We repeatedly subdivide this cube into $2^d, 2^{2d}, \ldots, 2^{pd}$ regular subcubes, and we denote the resulting cubes by C^ℓ_j where ℓ indicates the level of refinement and the multiindex $j \in \mathbb{N}^d$ refers to their positions as illustrated below:*

$$C^\ell_j \;:=\; \mathcal{I}^\ell_{j_1} \times \cdots \times \mathcal{I}^\ell_{j_d} \qquad \textit{with} \quad \mathcal{I}^\ell_i := \big[(i-1)2^{-\ell}h_{\max}, i2^{-\ell}h_{\max}\big)\,.$$



The sons (successors) $S(C^\ell_j)$ of a cube C^ℓ_j are defined as the 2^d cubes on level $\ell+1$ that are contained in C^ℓ_j. This hierarchy of cubes is then used to define a cluster tree with $\sum_{j=0}^p 2^{dj}$ vertices corresponding to the subcubes where for each index subset $t \in T_I^{(\ell)}$ the corresponding Chebyshev centres m_i, $i \in t$, will be included in exactly one cube $C^\ell_j =: C_t$. Let $\operatorname{root}(T_I) := I$ with corresponding cube $C^0_{(1,\dots,1)}$. For a vertex $t \in T_I$ with corresponding cube C_t, the set $S(t)$ of successors is defined by

$$S(t) := \begin{cases} \{s_C \mid C \in S(C_t)\} \setminus \{\emptyset\} & : \ \#t > n_{\min} \\ S(t) = \emptyset & : \ \textit{otherwise} \end{cases} \qquad \textit{where} \quad s_C := \{i \in t \mid m_i \in C\}.$$

Remark 10 *The above geometrically balanced construction can lead to a vertex having exactly one successor if, e.g., all Chebyshev centres of basis functions in $C^0_{(1,1)}$ are also in $C^1_{(1,1)}$ (see illustration above). If each cube on the finest level contains at least one Chebyshev centre, then Construction 9 will yield a cluster tree where each node is either a leaf or has exactly 2^d successors.*

3.2 Admissibility by Bounding Boxes

In Section 2.2 we have imposed the admissibility condition

$$\min\{\operatorname{diam}(B_t), \operatorname{diam}(B_s)\} \le \eta \operatorname{dist}(B_t, B_s)$$

for axially parallel boxes B_t, B_s containing the supports Ω_t, Ω_s of the respective basis functions (4).

Since the cube C_t that corresponds to the index set t contains only the Chebyshev-centres of Ω_i for $i \in t$ (but possibly $\Omega_i \not\subset C_t$), the bounding box for the cluster must be larger than the associated cube, and to guarantee $\Omega_t \subset B_t$ we define the local meshwidth h_t and bounding box B_t depending on a parameter $\rho \ge 1$ by

$$h_t := \max_{i \in t} \operatorname{diam}(\Omega_i), \qquad B_t := C_t + \frac{\rho}{2}\left[-h_t, h_t\right]^d. \tag{8}$$

Lemma 11 *For any two nodes $t, s \in T_I^{(\ell)}$ there holds*

$$\begin{aligned} \Omega_t &\subset B_t \\ \operatorname{diam}(B_t) &= \sqrt{d}(2^{-\ell} + \rho h_t) h_{\max} \qquad (9) \\ \operatorname{dist}(B_t, B_s) &\ge \operatorname{dist}(C_t, C_s) - \frac{\rho}{2}\sqrt{d}(h_t + h_s). \qquad (10) \end{aligned}$$

Proof: Let $i \in t$. By Construction 9 we get $m_i \in C_t$. The elements in the support Ω_i have a distance of at most $\frac{1}{2}\operatorname{diam}(\Omega_i)$ from the centre m_i which implies $\Omega_i \subset B_t$. The second and third part follow from the definition of B_t. ■

Using axially parallel bounding boxes will allow us to use a tensor product interpolation scheme for the approximation of the kernel function [1]. The parameter ρ has been introduced to enlarge the bounding boxes such that a later insertion of indices (arising from adaptive mesh refinement) does not require an update of the bounding box (see Section 4).

3.3 Construction of the Block Cluster Tree $T_{I\times I}$

Based on the cluster tree T_I from Construction 9 and on the admissibility condition (5) we define the block cluster tree $T_{I\times I}$ as follows.

Construction 12 (Canonical block cluster tree) *Let the cluster tree T_I be given. We define the block cluster tree $T_{I\times I}$ by* $\mathrm{root}(T) := I \times I$ *and for each vertex $t \times s \in T$ the set of successors*

$$S(t\times s) := \begin{cases} \emptyset & \text{if } \min\{\#t, \#s\} \le n_{\min} \text{ or } t\times s \text{ admissible} \\ \{t'\times s' \mid t' \in S(t), s' \in S(s)\} & \text{otherwise.} \end{cases}$$

Construction 9 followed by Construction 12 yields a block cluster tree $T_{I\times I}$, and its leaves determine the block partition used for an $\mathcal{H}$-matrix. The estimates for the storage requirements and matrix-vector multiplication complexity for this $\mathcal{H}$-matrix depend on the depth (see Definition 5) and the sparsity C_{sp} of the underlying block $\mathcal{H}$-tree $T_{I\times I}$.

Definition 13 (Sparsity) *Let $T_{I\times I}$ be a block cluster tree. We define the* sparsity (constant) C_{sp} *of $T_{I\times I}$ by*

$$C_{\mathrm{sp}} := \max_{t\in T_I} \#\{s \in T_I \mid t\times s \in T_{I\times I}\}. \tag{11}$$

Up to now we have not posed any condition on the locality of the supports of the basis functions φ_i, which is necessary for a successful $\mathcal{H}$-matrix construction. If the basis φ_i is refined adaptively, we cannot assume a uniform meshwidth h. Instead, we assume the following locality of the supports which, however, does not impose a severe restriction to a typical adaptive refinement scheme.

Assumption 1 (Locality) *We assume that the supports are locally separated in the sense that there exist two constants C_{sep} and $n_{\min}$ such that*

$$\max_{i\in I} \#\{j \in I \mid \mathrm{dist}(\Omega_i, \Omega_j) \le C_{\mathrm{sep}}^{-1}\mathrm{diam}(\Omega_i)\} \le n_{\min}. \tag{12}$$

The left-hand side is the maximal number of basis functions with 'relatively close' supports (see Figure 2). Note that the bound $n_{\min}$ is the same as in Definition 8, i.e., the choice of $n_{\min}$ has to satisfy (12).

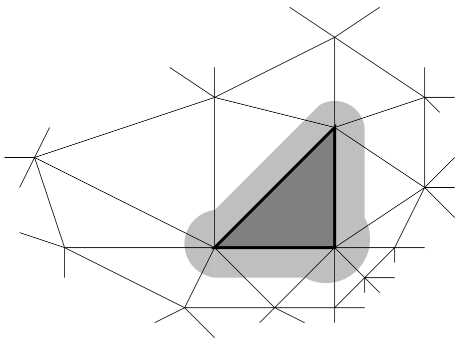


Figure 2: The triangle Ω_i under consideration is dark grey. The area with a distance of $C_{\mathrm{sep}}^{-1}\mathrm{diam}(\Omega_i)$ is light grey ($C_{\mathrm{sep}} := 4\sqrt{2}$). Here, 15 triangles (including Ω_i) are 'rather close' to Ω_i.

The next lemma is a generalisation of Lemma 4.5 in [7] where the respective estimates were obtained for an admissibility condition based upon the actual supports Ω_t instead of the (enlarged) bounding boxes B_t.

Lemma 14 *Let Assumption 1 be satisfied, and let $T_{I\times I}$ be the block cluster tree obtained from Constructions 9 and 12. Let $h_{\min} := \min_{i\in I} \operatorname{diam}(\Omega_i)$. Then the following statements hold:*
(a) All leaves $t \times s \in \mathcal{L}(T_{I\times I})$ are either admissible with respect to (5) or $\min\{\#t, \#s\} \le n_{\min}$.
(b) The depth of the tree is bounded by

$$\operatorname{depth}(T_{I\times I}) \le 1 + \log_2\left((1+\rho C_{\mathrm{sep}})\sqrt{d}h_{\max}h_{\min}^{-1}\right).$$

(c) The sparsity constant is bounded by

$$C_{\mathrm{sp}} \le \left(4 + 8\eta^{-1}\left(\sqrt{d}(1+\rho C_{\mathrm{sep}}) + \eta\rho C_{\mathrm{sep}}d\right)\right)^d.$$

Proof: (a) holds by Construction 12. **(b)**: let $t \in T_I^{(\ell)}$ be a non-leaf node. Then $\#t > n_{\min}$ such that

$$\operatorname{diam}(B_t) \overset{(9)}{=} \sqrt{d}(2^{-\ell} + \rho h_t)h_{\max} \overset{(12)}{\le} \sqrt{d}(2^{-\ell} + \rho C_{\mathrm{sep}}\operatorname{diam}(C_t))h_{\max} = \sqrt{d}(1+\rho C_{\mathrm{sep}})2^{-\ell}h_{\max} \tag{13}$$

while $\operatorname{diam}(B_t) \ge \operatorname{diam}(\Omega_i) \ge h_{\min}$. This yields $2^\ell \le \sqrt{d}(1+\rho C_{\mathrm{sep}})h_{\max}h_{\min}^{-1}$.
(c): we exploit the structure of the regular subdivision of $[0, h_{\max})^d$ into the cubes C_t. Let $t \in T_I^{(\ell)}$ be a cluster with $\#t > n_{\min}$. The number of cubes C_s on level ℓ that touch C_t is at most 3^d. By induction it follows that the number of cubes on level ℓ with a distance less than $j2^{-\ell}h_{\max}$ to C_t is bounded by $(1+2j)^d$. Let $s \in T_I^{(\ell)}$ with $\#s > n_{\min}$ and $\operatorname{dist}(C_t, C_s) > j2^{-\ell}h_{\max}$. The diameters of the respective bounding boxes and their distance can be estimated by

$$\begin{aligned}
\operatorname{diam}(B_t) &\overset{(13)}{\le} \sqrt{d}(1+\rho C_{\mathrm{sep}})2^{-\ell}h_{\max},\\
\operatorname{dist}(B_t, B_s) &\overset{(10)}{\ge} \operatorname{dist}(C_t, C_s) - \frac{\rho}{2}\sqrt{d}(h_t + h_s)\\
&> j2^{-\ell}h_{\max} - \frac{\rho}{2}\sqrt{d}C_{\mathrm{sep}}(\operatorname{diam}(C_t) + \operatorname{diam}(C_s))\\
&= j2^{-\ell}h_{\max} - \rho C_{\mathrm{sep}}d2^{-\ell}h_{\max}.
\end{aligned}$$

If $t \times s$ is not admissible, then (5) yields the estimate

$$\sqrt{d}(1+\rho C_{\mathrm{sep}})2^{-\ell}h_{\max} > \eta(j2^{-\ell}h_{\max} - \rho C_{\mathrm{sep}}d2^{-\ell}h_{\max})$$

which implies

$$j < \eta^{-1}\left(\sqrt{d}(1+\rho C_{\mathrm{sep}}) + \eta\rho C_{\mathrm{sep}}d\right) =: j_{\max}.$$

Therefore, the number of clusters $s \in T_I^{(\ell)}$ for which $t \times s$ is not admissible is bounded by $(1+2j_{\max})^d$. The number of successors of s is bounded by 4^d such that on level $\ell+1$ there are at most $4^d(1+2j_{\max})^d$ cluster s' with $t' \times s' \in T_{I\times I}$ for any son t' of t. ■

The estimates in Lemma 14 show that the sparsity of the tree is independent of the cardinality of I. In practice the sparsity should be smaller than 100 while the depth of the tree is typically proportional to $\log(\#I)$. In the following section we will show that 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 $T_{I\times I}$.

3.4 Data-Sparsity of the $\mathcal{H}$-matrix Format

In this section we estimate the storage requirements of an $\mathcal{H}$-matrix and the complexity of the matrix-vector multiplication based on the *sparsity* C_{sp} (11) of the block cluster tree $T_{I\times I}$.

Lemma 15 (Storage, Lemma 2.4 in [7]) *Let $T_{I\times I}$ be a block cluster tree based on T_I with sparsity constant C_{sp} (11) and minimal block size $n_{\min}$. Then the storage requirements $N_{\mathcal{H},St}(T_{I\times I},k)$ for an $\mathcal{H}$-matrix $M \in \mathcal{H}(T_{I\times I},k)$ are bounded by*

$$N_{\mathcal{H},St}(T_{I\times I},k) \le 2(1+\mathrm{depth}(T_{I\times I}))C_{\mathrm{sp}}\max\{k,n_{\min}\}\#I$$

where bounds for the depth and sparsity, in turn, are given in Lemma 14.

Lemma 16 (Matrix-Vector Product) *Let $T_{I\times I}$ be a block cluster tree. The complexity $N_{\mathcal{H}\cdot v}(T_{I\times I},k)$ of the matrix-vector product in the set of $\mathcal{H}$-matrices can be bounded from above and below by*

$$N_{\mathcal{H},St}(T_{I\times I},k) \le N_{\mathcal{H}\cdot v}(T_{I\times I},k) \le 2N_{\mathcal{H},St}(T_{I\times I},k).$$

Proof: According to Remark 3 there holds

$$N_{R,St} \le N_{R\cdot v} \le 2N_{R,St}, \qquad N_{F,St} \le N_{F\cdot v} \le 2N_{F,St}.$$

The proposition follows since an $\mathcal{H}$-matrix consists blockwise of full matrices and $R(k)$-matrices. ■

4 Adaptive Refinement and Clustering

In the previous section we described a method for the data-sparse approximation of the stiffness matrix arising in the Galerkin discretisation of an integral operator with asymptotically smooth kernel function. Given a fixed set of basis functions φ_i,

$$\mathcal{B}_n := \{\varphi_1,\dots,\varphi_n\}, \tag{14}$$

which span the n-dimensional space V_n, we seek an approximation $u_n \in V_n$ to the solution u of the equation

$$\int_\Omega g(x,y)u(y)\mathrm{d}y = f(x). \tag{15}$$

The goal is to find a small $n \in \mathbb{N}$ such that the approximation error $\|u-u_n\|$ in a suitable norm is smaller than a desired (given) accuracy $\varepsilon > 0$. In order to achieve this, we start with a small set $\mathcal{B}_{n_0}$ of basis functions, compute a coarse approximation u_{n_0}, estimate the approximation error $\|u-u_{n_0}\|$ by some (local) error estimator (see, e.g., [4, 13]) and replace the set $\mathcal{B}_{n_0}$ by a larger set of basis functions $\mathcal{B}_{n_1} := \{\varphi_1,\dots,\varphi_{n_1}\}$. This process is repeated until the error estimator gives a reliable upper bound $\bar{\varepsilon}$ that is below the desired accuracy ε.

Each step of this adaptive scheme yields a linear system of equations involving the matrix G_{n_j} (3). For a matrix given in full matrix representation, the removal of n' and addition of n'' entries require n' rows and columns of G_{n_j} to be removed while n'' rows and columns have to be added and their respective entries have to be computed. All the previously computed entries can be retained. If the matrix G_{n_j} is given in an $\mathcal{H}$-matrix representation, we might be forced to reassemble most of the matrix entries if a cardinality balanced cluster tree has been used to determine the $\mathcal{H}$-matrix partition (see Example 17). The geometrically regular Construction 9 has been designed to avoid this drawback (as will be illustrated in the following sections).

Example 17 (The problem of a cardinality balanced tree) *For simplicity, we consider a uniform grid on the unit interval $[0,1]$ with $n = 2^p$ piecewise constant basis functions and corresponding index set $I = \{1,\dots,n\}$. Let the tree T_I be constructed in a cardinality balanced way (cf. [7]) with the two clusters $\{1,\dots,n/2\},\{n/2+1,\dots,n\}$ on the first level. If the index set I is extended by two indices to $I' = \{1,\dots,n+1,n+2\}$ (subdividing the two rightmost basis functions each into two parts), then the clusters on the first level are $\{1,\dots,n/2+1\},\{n/2+2,\dots,n+1,n+2\}$ which are both different from the clusters on the first level of the original tree T_I, and the resulting clusters on the higher levels of the cardinality balanced tree will almost all be different from the clusters in T_I. This will result in an $\mathcal{H}$-matrix for the index set $I' = \{1,\dots,n+1,n+2\}$ with a completely different block partition than the one for I such that most entries have to be recomputed instead of being copied over from the $\mathcal{H}$-matrix obtained for I.*

In the following subsections, we show that a geometrically balanced clustering as in Construction 9 leads to $\mathcal{H}$-matrix partitions where previously computed entries can be copied over whereas only entries that correspond to the refined set of basis functions have to be computed for the update.

Remark 18 (adaptively expandable $\mathcal{H}$-matrices) *If, as for this paper, we store the two factors A, B of an $R(k)$-matrix $M = AB^T$ of size $\mathcal{O}(n)$ as arrays, then the expansion by one index requires us to reallocate the arrays and to copy the entries. If we use linked lists for the full matrices A, B, then the detection of a row or column corresponding to an index i would require to run through the whole list of indices. In both cases, the removal or addition of one index results in a complexity of $\mathcal{O}(n)$. In [6], we developed adaptively expandable $\mathcal{H}$-matrices where each degree of freedom stores the information about its location in the matrix blocks. This leads to an optimal complexity with respect to storage allocation and data copying. In practice, however, it is the computation of the matrix entries which is the bottleneck, whereas the storage allocation and data copying are negligible.*

In Sections 4.1 - 4.2, we consider for simplicity the necessary updates of the cluster tree, the block cluster tree and the matrix entries when only a single index is added. In Section 4.3, we then provide details on an implementation that includes the addition or removal of several indices (as a result of an adaptive mesh refinement or coarsening).

4.1 Updating the Cluster Tree T_I

The creation of the cluster tree T_I described in Construction 9 depends exclusively on the Chebyshev centres m_i of the supports of the basis functions φ_i. Let $\mathcal{B}_n := \{\varphi_1, \ldots, \varphi_n\}$ denote the set of basis functions with index set $I = \{1, \ldots, n\}$ and an already constructed (geometrically regular) cluster tree T_I. We now describe how the cluster tree T_I will be updated to a cluster tree $T_{I'}$ for the index set $I' := \{1, \ldots, n+1\}$ when a single basis function φ_{n+1} is added to the basis.

On each level $\ell = 0, \ldots, \mathrm{depth}(T_I)$, the Chebyshev centre m_{n+1} of the new basis function φ_{n+1} is contained in at most one of the cubes C_t that corresponds to one of the clusters $t \in T_I^{(\ell)}$. We define the nodes of the tree $T_{I'}$ for the index set I' as

$$t' := \begin{cases} t \cup \{n+1\} & \text{if } m_{n+1} \in C_t \\ t & \text{otherwise.} \end{cases}$$

In addition, we have to distinguish the following three cases to decide whether further nodes have to be added. For this purpose, let t be the node in T_I with the highest level number ℓ which is enlarged by the index $n+1$, i.e., t is the node with the highest level number ℓ such that C_t contains the Chebyshev centre m_{n+1} of the new basis function φ_{n+1}.

- If the cluster t is a leaf with a blocksize smaller that n_{min}, then no further nodes have to be added.
- If the cluster t is a leaf with blocksize n_{min}, then the cluster t' has to be further subdivided according to Construction 9, see Figure 3.
- If the node t is not a leaf (since m_{n+1} is contained in a cube $C_j^{\ell+1}$ on the next finer level $\ell + 1$ that contains no other Chebyshev centre m_i, $i = 1, \cdots, n$), then t' will have a new successor in $T_{I'}$ (not present in T_I) that contains only the index $n+1$.

Similarly, when an index is removed from I to obtain $I' := \{1, \ldots, n-1\}$, the cluster tree for I' is obtained from T_I by eliminating the index n from all nodes of T_I. If the father of the leaf that contains the index n has cardinality $n_{min} + 1$, then its subdivision has to be undone.

In practice we will typically remove and/or insert several indices at once after a refinement of the grid. The cost to remove or insert a single index in the tree T_I is $\mathcal{O}(\mathrm{depth}(T_I))$. Therefore, the adaptive construction of the tree T_I (inserting the indices one by one) has the same complexity as Construction 9.

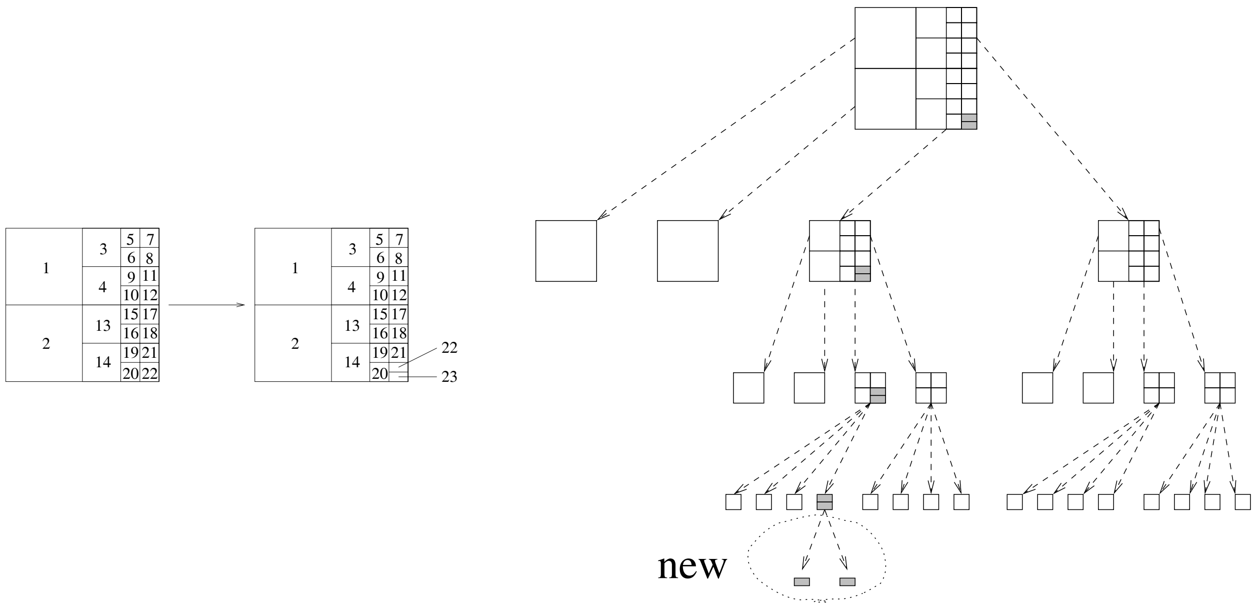


Figure 3: Left: the grid (constant basis functions on each panel) is refined by replacing one of the panels by the two dark panels. Right: in the tree T_I the two new panels are inserted in one cluster on each level. The area marked “new” originates because $n_{\min} = 1$.

4.2 Updating the Block Cluster Tree $T_{I\times I}$ and the Matrix Entries

The next step is to update the block cluster tree $T_{I\times I}$ according to the changes in the tree T_I. Here, we again assume that the index set I is enlarged (or reduced) to the set I' only by a single index.

The block cluster trees $T_{I\times I}$ and $T_{I'\times I'}$ are defined in a canonical way according to Construction 12. Since in the tree T_I only $q = \mathcal{O}(\mathrm{depth}(T_I))$ nodes were changed, we have to consider at most $C_{\mathrm{sp}}q$ nodes in the tree $T_{I\times I}$ that differ from the ones in the tree $T_{I'\times I'}$ (cf. Definition 11 and see Figure 4). The update of the

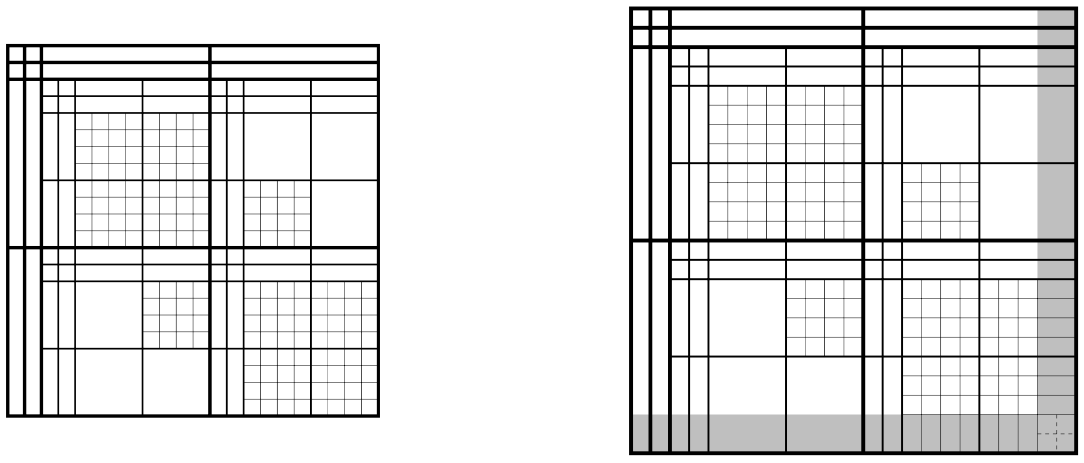


Figure 4: Left: the block cluster tree $T_{I\times I}$ corresponding to the cluster tree T_I in Figure 3. Right: in the tree $T_{I'\times I'}$ the shaded parts differ from $T_{I\times I}$.

block cluster tree $T_{I\times I}$ is done along with the update of the matrix entries. In the optimal case (a minimum of work) only the entries corresponding to the row and column of the new index have to be assembled (the shaded parts in Figure 4) while all previously computed entries can be kept. This optimum is typically not reached, but the cases where we have to reassemble whole blocks are rare. The parameter ρ in the definition of the bounding box (8) enables us to reduce their number. A modified admissibility condition would make a reassembly completely unnecessary (but has other drawbacks).

Remark 19 (Modification of the admissibility) *For theoretical purposes one could use the admissibility condition*

$$\min\{\mathrm{diam}(C_t), \mathrm{diam}(C_s)\} \leq (1 + C_{\mathrm{sep}} + \eta C_{\mathrm{sep}})^{-1}\eta \mathrm{dist}(C_t, C_s)$$

instead of (5), or in other words replace η by $\tilde{\eta} := (1 + C_{\mathrm{sep}} + \eta C_{\mathrm{sep}})^{-1}\eta$ and use the admissibility for the regular cubes. Since $(1 + C_{\mathrm{sep}} + \eta C_{\mathrm{sep}})$ is a constant, the statements of Lemma 14 can even be simplified and the bounding boxes fulfil the standard admissibility (5) for the parameter η. The advantage is that the admissibility and the bounding boxes are independent of the basis functions φ_i (as long as they fulfil the local separability with the constant C_{sep}). Therefore, admissibility of blocks does only change if the insertion or removal of indices breaks the minimal blocksize $n_{\min}$. As a result of the more restrictive admissibility however, the sparsity C_{sp} is typically increased which is not desirable in practice.

For each of the nodes $t \times s \in T_{I\times I}$ that need to be updated since $t \in T_I$ and/or $s \in T_I$ are updated to $t' \in T_{I'}$ and/or $s' \in T_{I'}$, resp., we have to distinguish four cases:

1. $t \times s$ and $t' \times s'$ correspond to $R(k)$-matrix blocks (see Subsection 4.2.1).
2. $t \times s$ and $t' \times s'$ correspond to full matrix blocks (see Subsection 4.2.2).
3. $t \times s$ corresponds to a full matrix block whereas $t' \times s'$ does not, or vice-versa (see Subsection 4.2.3).
4. $t \times s$ corresponds to an $R(k)$-matrix block whereas $t' \times s'$ does not, or vice-versa (see Subsection 4.2.4).

For simplicity, we will consider blocks $t \times s$ and $t' \times s$ where the nodes t, t' differ only by one index. In practice, the removal and addition of several indices is performed in a single step; an algorithm that performs the update of the trees T_I and $T_{I\times I}$ as well as the matrix entries for a set of indices will be presented in Section 4.3.

4.2.1 Updating $R(k)$-matrix Blocks

Let the blocks $t \times s \in T_{I\times I}$ and $t' \times s \in T_{I\times I}$ be admissible and large enough (i.e., $\geq n_{min}$) such that the corresponding blocks M, M' of the stiffness matrix G from (3) are stored as $R(k)$-matrices:

$$\begin{aligned} M_{ij} &\overset{(6)}{=} \int_\Omega \int_\Omega \varphi_i(x) \sum_{\nu\in\{0,\dots,m\}^d} g(x_\nu, y) L_\nu(x) \varphi_j(y) \,\mathrm{d}x \,\mathrm{d}y \\ &= \sum_{\nu\in\{0,\dots,m\}^d} \underbrace{\int_\Omega \varphi_i(x) L_\nu(x) \,\mathrm{d}x}_{A_{i\nu}} \underbrace{\int_\Omega g(x_\nu, y)\varphi_j(y) \,\mathrm{d}y}_{B_{j\nu}} \\ &= \sum_{\nu\in\{0,\dots,m\}^d} A_{i\nu} B_{j\nu}. \end{aligned}$$

If t' is obtained after removing an index from t, i.e., $t = t' \dot{\cup} \{\mu\}$, then the matrix M' is the restriction of M to the subblock $t' \times s$, i.e.

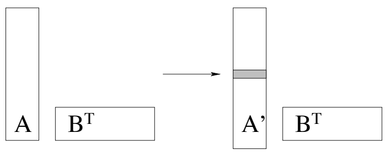
$$M'_{ij} = M_{ij} \quad \text{for all } (i, j) \in t' \times s,$$

and the entries of the matrix A only have to be copied into the respectively smaller matrix A'.

If t' is obtained from adding an index $\mu \in I'$ to t, i.e., $t' = t \dot{\cup} \{\mu\}$, we need to distinguish the case where the bounding box $B_{t'}$ is included in B_t from the case where $B_{t'} \not\subseteq B_t$ since the polynomial L_ν in (6) depends on the bounding box B_t.

In the first case, $B_{t'} \subseteq B_t$, the matrix M' is given by

$$\begin{aligned} M' &= A'B^T, \\ A'_{i\nu} &= \begin{cases} A_{i\nu} & \text{if } i \in t \\ \int_\Omega \varphi_i(x) L_\nu(x)\,\mathrm{d}x & \text{if } i = \mu, \nu \in \{0,\dots,m\}^d, \end{cases} \end{aligned}$$



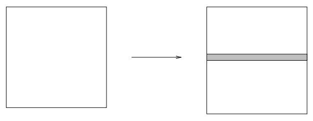
i.e., we have to compute $k = (m+1)^d$ new entries.

In the second case, $B_{t'} \not\subseteq B_t$, we compute an $R(k)$-matrix approximation with a bounding box $B_{t'}$ for an increased parameter ρ (e.g., $\rho_{new} = 2 \cdot \rho_{old}$). However, since we expect the supports to shrink in the context of adaptive *refinement* this case is unlikely to occur.

The key observation is that the entries of the two factors A, B in the $R(k)$-matrix representation of admissible matrix blocks can be computed independently of each other.

4.2.2 Updating Full Matrix Blocks

Let the blocks $t \times s \in T_{I\times I}$ and $t' \times s \in T_{I\times I}$ be inadmissible leaves so that the corresponding blocks M, M' of the stiffness matrix G are stored as full matrices. In this case the addition or removal of the index μ corresponds to the addition or removal of one row of the matrix and all other entries can be kept.



4.2.3 Subdivision of Inadmissible Leaves

Let the block $t \times s \in T_{I\times I}$ be an inadmissible leaf with $\#t = n_{\min}$, $\#s > n_{\min}$, and an index is added to t to obtain $t' = t \cup \{\mu\}$ such that $t' \times s$ has to be subdivided. Then the following two cases may occur:

Case 1: Inadmissible descendants of $t' \times s$ that are leaves are stored as full matrices and the previously assembled entries of the block $t \times s$ (which was stored as full matrix) can be copied.

Case 2: Admissible descendants of $t' \times s$ that are leaves of the form $t' \times s_\ell$ are stored as $R(k)$-matrices, and a low rank approximation for the block $t' \times s_\ell$ has to be computed for bounding boxes with parameter $\rho := 2$. The number of newly computed entries of the entire block is $k(\#t' + \#s_\ell) = k(n_{\min} + 1 + \#s_\ell)$ which is just a constant $(n_{\min} + 1)$ times more than the average number of entries per row $(k(\#t' + \#s_\ell)/\#t')$.

4.2.4 Subdivision of Admissible Blocks

Let the block $t \times s \in T_{I\times I}$ be an admissible leaf and let $t' = t \cup \{\mu\}$ such that $t' \times s$ is inadmissible and has to be subdivided. This case might occur when the bounding box $B_{t'}$ is not included in B_t such that the admissibility condition is now violated for $t' \times s$.

Case 1: Descendants $t_\ell \times s_\ell$ of $t' \times s$ where t_ℓ does not contain the index μ are admissible with the respective bounding boxes B_t, B_s. The corresponding matrix block is a submatrix of the respective one from the block $t \times s$ and we inherit the bounding box.

Case 2: Descendants $t'_\ell \times s_\ell$ of $t' \times s$ that are leaves and where t'_ℓ contains the index μ have to be newly assembled (cannot inherit the bounding box from t because Ω_μ is not contained in it), either as full matrices or as $R(k)$-matrices.

Again, we point out that in the context of adaptive *refinement* we do not expect the block $t' \times s$ to be inadmissible (if $t \times s$ was admissible), since the support Ω_μ is supposed to shrink and be smaller than the supports Ω_i of the basis functions φ_i that correspond to the indices $i \in t$.

If, on the other hand, an index is removed from t to obtain t', then $t' \times s$ might be admissible even though $t \times s$ was not admissible and therefore further subdivided. In this case, an $R(k)$-matrix representation for the block $t' \times s$ has to be computed.

4.3 Implementation of the Update

In this section we present the algorithms for the update of the cluster tree T_I, the block cluster tree $T_{I\times I}$ and the matrix G from (3). Here, the index set I, the corresponding trees and the stiffness matrix are given. The indices in the set $I_{\text{out}} \subset I$ are to be removed from the trees while the indices in I_{in} have to be inserted yielding the new index set

$$I' = (I \setminus I_{\text{out}}) \,\dot\cup\, I_{\text{in}}.$$

Construction 20 (Update of the cluster tree T_I) *We construct the new cluster tree $T_{I'}$ from the root to the leaves as follows:*

1. *Remove all indices from I_{out} by $t' := t \setminus I_{\text{out}}$ for all $t \in T_I$. Mark all changed clusters t' by a flag. Empty clusters may appear.*
2. *Add the new indices from I_{in} by $t' := t' \cup t''$, $t'' := \{i \in I_{\text{in}} \mid m_i \in C_t\}$. If $\#t'' > 0$ then mark t'.*
3. *Subdivide the leaves $t' \in \mathcal{L}(T_{I'})$ with $\#t' > n_{\min}$, mark these new clusters and discard all empty clusters.*

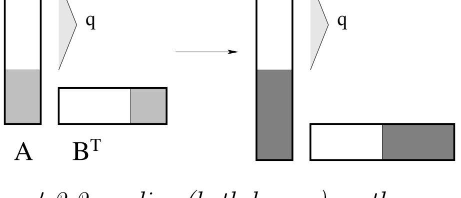
If the clusters are stored as an array, it is advantageous to store the retained indices in the first part of the array and the new (removed) indices in the second part of the array. A counter q for the number of retained indices allows an easy access to the second part:

$$t = [\ \underbrace{i_1, \dots, i_q}_{\text{kept indices}}\ ,\ \underbrace{i_{q+1}, \dots, i_{\#t}}_{\text{removed indices}}\ ] \qquad t' = [\ \underbrace{i_1, \dots, i_q}_{\text{kept indices}}\ ,\ \underbrace{i_{q+1}, \dots, i_{\#t'}}_{\text{new indices}}]$$

Construction 21 (Update of the block cluster tree $T_{I'\times I'}$ and matrix) *The update of the block cluster tree $T_{I'\times I'}$ and the hierarchical matrix is done from the root $I' \times I'$ to the leaves blockwise for each $t \times s \in T_{I\times I}$ with corresponding block $t' \times s' \in T_{I'\times I'}$:*

1. *If $t \times s = t' \times s'$ then nothing is to be done (t', s' unmarked).*
2. *If $t \times s \neq t' \times s'$ and both are not leaves: update the number of successors of $t \times s$ to those of $t' \times s'$. Since matrix entries are only stored for each leaf of the tree, nothing has to be assembled.*
3. *If $t \times s \neq t' \times s'$ and at least one of the two is a leaf: update the number of successors of $t \times s$ to those of $t' \times s'$. We distinguish three cases:*
   - *(a) Both are admissible so that the update from Section 4.2.1 applies. First we check whether the old bounding box is still valid (contains the supports of the basis functions). If not, the block has to be reassembled with parameter $\rho := 2$ for the bounding box. Otherwise, we define*

$$\begin{aligned} M' &= A'B^T, \\ A'_{i\nu} &= \begin{cases} A_{i\nu} & \text{if } i \in t \\ \int_\Omega \varphi_i(x) L_\nu(x)\,\mathrm{d}x & \text{otherwise.} \end{cases} \\ B'_{i\nu} &= \begin{cases} B_{i\nu} & \text{if } i \in t \\ \int_\Omega \varphi_i(x) g(x_\nu, y)\,\mathrm{d}y & \text{otherwise.} \end{cases} \end{aligned}$$



   - *(b) Both are inadmissible so that either the update from Section 4.2.2 applies (both leaves) or the new block has to be reassembled in parts (cf. Section 4.2.3).*
   - *(c) One of the two is admissible, the other one not. Then the new block is reassembled.*

The result of the previous constructions is the stiffness matrix G for the refined grid using the geometrically balanced (regular) clustering. The matrix format is not changed at all, the $\mathcal{H}$-matrix arithmetic established in [7] can be applied for the purpose of preconditioning or solving the discrete system.

5 Numerical Results

In our numerical test we illustrate the efficiency of our update procedure for (local) grid refinement. The model problem that we consider is

$$\mathcal{G}[u](x) = f(x), \qquad x \in \Gamma := \partial\Omega, \quad \Omega := [-1,1]^3$$

where $u|_\Gamma$ are the Dirichlet data of a harmonic function u in the domain Ω, $\mathcal{G}$ is the double layer potential operator

$$\mathcal{G}[u](x) := \frac{1}{2}u(x) + \frac{1}{4\pi}\int_\Gamma \frac{\langle n(y), x-y\rangle u(y)}{\|x-y\|^3}\,\mathrm{d}\Gamma_y$$

and the right-hand side is given by the single layer potential operator $\mathcal{V}$ applied to the Neumann data $f := \mathcal{V}\partial_n u$,

$$\mathcal{V}[u](x) := \frac{1}{4\pi}\int_\Gamma \frac{\partial_n u(y)}{\|x-y\|}\mathrm{d}\Gamma_y.$$

The harmonic function u is

$$u(x) := \frac{1}{\|x-x_0\|}, \qquad x_0 := (1,1,1.001).$$

For the discretisation in the $\mathcal{H}$-matrix format we choose $n_{\min} := 32$ for the minimal cluster size, $\eta := 2.0$ in the admissibility condition and $k := 8$ for the blockwise rank.

Remark 22 (Update for full matrices and $\mathcal{H}$-matrices) *For an $n \times n$ matrix in full matrix representation the addition of n_{new} degrees of freedom means that $2n_{\text{new}}n - n_{\text{new}}^2$ entries have to be newly assembled, i.e., p% new degrees of freedom result in $(2p - p^2/n)$% new entries. This is different for $\mathrm{R}(k)$-matrices. There, p% new degrees of freedom result in p% new entries. Therefore, $\mathcal{H}$-matrices, which consist blockwise of $\mathrm{R}(k)$-matrices, allow for an even more efficient update than full matrices. If p% degrees of freedom are new then we expect between p% and $2p$% of the matrix entries to be newly assembled while the others are kept.*

In the adaptive scheme we refine 1%, 5% or 25% of the three-dimensional grid where the discretisation error is the largest, such that roughly 2%, 10% or 50% of the degrees of freedom on the refined grid are new.

The time for an adaptive update is compared to the time it takes to reassemble the whole stiffness matrix. The starting grid has $n_0 = 3072$, $n_0 = 12288$ or $n_0 = 49152$ degrees of freedom and is a uniform triangulation of $\Gamma := \partial\Omega$. The interpolation scheme is linear ($m = 1$) and for the nearfield entries (entries in inadmissible leaves) we use Gaussian quadrature with $q = 2$ points per axis.

In Figure 5 we have visualised the update of a small $\mathcal{H}$-matrix. One can see that the reassembled blocks are just the ones corresponding only to new degrees of freedom, while the rest is updated efficiently.

The numerical results are presented in Table 1 and were produced on a Sun UltraSparc III with 900 MHz CPU clock rate and 150 MHz memory clock rate. We conclude that for a moderate size of the grids the local update of p% of the grid can be done in almost p% of the time for the assembly of the stiffness matrix by use of the update procedure.

Acknowledgements

The authors thank Jelena Djokić from the “Max-Planck-Institut für Mathematik in den Naturwissenschaften” (Leipzig) for implementing the adaptive algorithms in the $\mathcal{H}$-matrix library HLib.

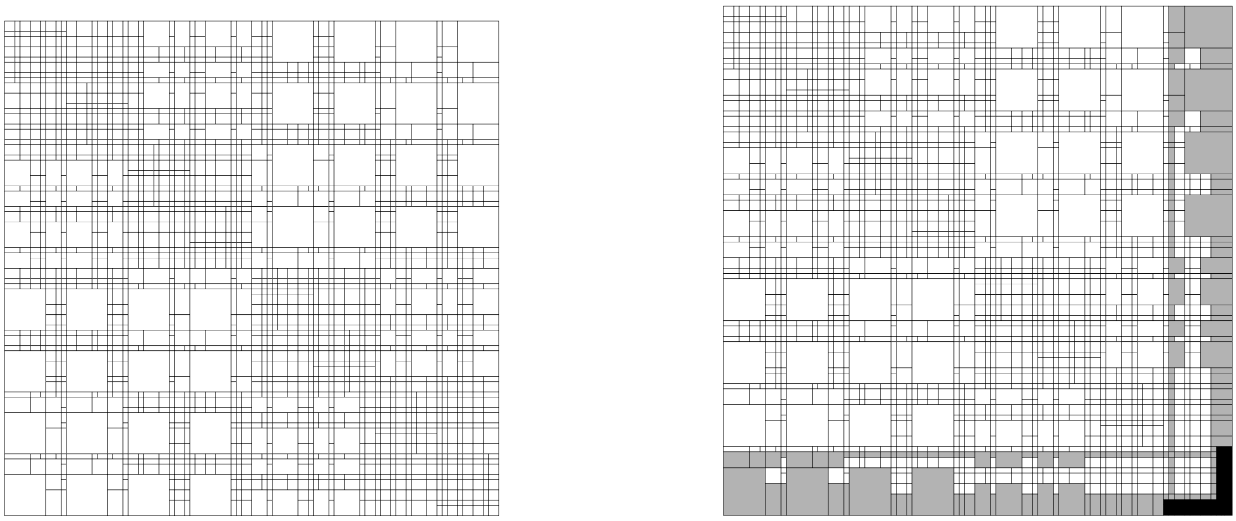


Figure 5: The stiffness matrix with $n_0 = 3072$ degrees of freedom (left) and the updated stiffness matrix with $n_1 = 3108$ degrees of freedom (right). The grey blocks are updated blocks of the matrix and the black blocks are reassembled parts.

$n_0 = 3072$	$n_1 = 3108$	$n_1 = 3254$	$n_2 = 3986$
new	2.3%	11.2%	45.9%
reassembly	5.45s	5.69s	6.99s
update	0.38s	1.14s	4.71s
relative	7.0%	20.0%	67.4%

$n_0 = 12288$	$n_1 = 12422$	$n_1 = 13002$	$n_1 = 15806$
new	2.2%	11.0%	44.5%
reassembly	29.5s	31.7s	40.8s
update	1.0s	5.8s	23.1s
relative	3.4%	18.3%	56.6%

$n_0 = 49152$	$n_1 = 49682$	$n_1 = 51880$	$n_1 = 62544$
new	2.1%	10.5%	42.8%
reassembly	169.0s	209.6s	252.7s
update	6.1s	31.8s	121.8s
relative	3.6%	15.2%	48.2%

Table 1: Time (in seconds) for the update of the $\mathcal{H}$-matrix compared to reassembly starting with $n_0 = 3072$, $n_0 = 12288$ and $n_0 = 49152$ degrees of freedom, resp.

References

[1] S. Börm, L. Grasedyck. *Low-rank approximation of integral operators by interpolation,* Preprint No. 72 (2002), Max-Planck-Institut für Mathematik in den Naturwissenschaften, Leipzig. To appear in *Computing.*

[2] S. Börm, L. Grasedyck, W. Hackbusch, *Introduction to hierarchical matrices with applications, Engineering Analysis with Boundary Elements*, 27:405–422, 2003.

[3] W. Dahmen, R. Schneider, *Wavelets on manifolds I: Construction and domain decomposition, SIAM J. of Math. Anal.*, 31:184–230, 1999.

[4] B. Faermann, *Localization of the Aronszajn-Slobodeckij norm and application to adaptive boundary element methods. Part II. The three-dimensional case, Numer. Math.* 92:467–499, 2002.

[5] L. Grasedyck. *Theorie und Anwendungen Hierarchischer Matrizen.* Doctoral thesis, University Kiel, Germany, 2001. (German)

[6] L. Grasedyck, W. Hackbusch, S. Le Borne. *Adaptive refinement and clustering of $\mathcal{H}$-matrices.* Preprint No. 106 (2001), Max-Planck-Institut für Mathematik in den Naturwissenschaften, Leipzig.

[7] L. Grasedyck, W. Hackbusch. *Construction and arithmetics of $\mathcal{H}$-matrices. Computing*, 70:295–334, 2003.

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

[9] W. Hackbusch, B. N. Khoromskij. *A sparse $\mathcal{H}$-matrix arithmetic. Part II: Application to multi-dimensional problems. Computing*, 64:21–47, 2000.

[10] W. Hackbusch, Z. P. Nowak. *On the fast matrix multiplication in the boundary element method by panel clustering. Numer. Math.*, 54:463–491, 1989.

[11] T. J. Rivlin. *The Chebyshev Polynomials.* Wiley-Interscience, 1984.

[12] V. Rokhlin. *Rapid solution of integral equations of classical potential theory. J. Comput. Phys.*, 60:187–207, 1985.

[13] H. Schulz, O. Steinbach. *A new a posteriori error estimator in adaptive direct boundary element method, Calcolo* 37:79–96, 2000.

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

Lars Grasedyck and Wolfgang Hackbusch
Max-Planck-Institute for Mathematics in the Sciences
Inselstr. 22-26
D-04103 Leipzig
Germany
lgr@mis.mpg.de, wh@mis.mpg.de

Sabine Le Borne
Department of Mathematics
Tennessee Technological University
Cookeville, TN 38505
USA
sleborne@tntech.edu