A Sparse Matrix Arithmetic based on $\mathcal{H}$-Matrices. Part I: Introduction to $\mathcal{H}$-Matrices[*]

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

Abstract

A class of matrices ($\mathcal{H}$-matrices) is introduced which have the following properties. (i) They are sparse in the sense that only few data are needed for their representation. (ii) The matrix-vector multiplication is of almost linear complexity. (iii) In general, sums and products of these matrices are no longer in the same set, but their truncations to the $\mathcal{H}$-matrix format are again of almost linear complexity. (iv) The same statement holds for the inverse of an $\mathcal{H}$-matrix.

This paper is the first of a series and is devoted to the first introduction of the $\mathcal{H}$-matrix concept. Two concrete formats are described. The first one is the simplest possible. Nevertheless, it allows the exact inversion of tridiagonal matrices. The second one is able to approximate discrete integral operators.

AMS Subject Classifications: 65F05, 65F30, 65F50.
Key words: Hierarchical matrices, hierarchical block partitioning, sparse matrices, matrix inversion.

1 Introduction

When dealing with linear systems of n equations, one has optimal efficiency if the computational amount of work is $O(n)$. For many situations characterised by a sparse system matrix A, one knows optimal solution algorithms. A basic problem arises for systems with non-sparse matrices. Since n^2 entries are to be used, the $O(n^2)$-complexity seems to be unavoidable (the example of the fast Fourier transform shows that this is not true in general). Two different techniques have been developed for full matrices. In the case of matrices obtained from integral operators, the panel clustering technique uses a representation with only $O(n \log n)$ data, which enables a matrix-vector multiplication with almost linear[1] complexity. Another approach is the matrix compression after using a special basis (e.g., wavelet bases) such that only few entries are of a non-negligible size.

In all the modern approaches to fast linear algebra applications, one tries to avoid matrix-matrix operations and uses only matrix-vector multiplications (or the vector-vector scalar product). In particular, the construction of an inverse matrix is forbidden. This is by two reasons; first, the computation is rather expensive ($O(n^3)$ in the general case) and, second, the inverse A^{-1} is usually a full matrix, even if A is sparse, and therefore the multiplication of A^{-1} by a vector is of $O(n^2)$-complexity.

A severe problem arises whenever the computation involves a Schur complement, since it contains the product $P := A * B^{-1} * C$ with given (sparse) matrices A, B, C. In this case, all actions must be manageable by performing the matrix-vector multiplication $P * x$ (involving the solution of $Bz = y := Cx$). There is no access to matrix entries of P, since the computation of B^{-1} is unattainable.

In this paper, we follow a completely different strategy. We do approximate full matrices, form (approximate) products of these matrices and compute the (approximate) inverse matrix. In the case of the Schur complement involving $A * B^{-1} * C$, we propose to compute this product directly. Surprisingly, the amount of work is almost linear in the dimension n. Obviously, this approach allows quite new applications. An idea which is rather close to our approach is described in the papers [1], [12], however, there the hierarchical treatment is missing, which is central in the present presentation.

[*]This paper has appeared in *Computing* **62** (1999) 89-108. This version contains minor corrections.
[1]Almost linear complexity means $O(n \log^k n)$, i.e., linear up to logarithmical factors.

The key point is the description of a class of matrices (called $\mathcal{H}$-matrices[2], where $\mathcal{H}$ abbreviates "hierarchical"). These matrices are not sparse in the sense that there are only few non-zero entries, but they are *data-sparse* in the sense that these matrices are described by only few data. The class of $\mathcal{H}$-matrices contains certain sparse matrices and approximates very well full matrices as they arise from integral operators or from the inversion of matrices corresponding to elliptic boundary value problems. The underlying idea is closely related to the panel clustering technique mentioned above (cf. [2], [3], [5], [7], [8], [9], [10], [11]).

In order not to overburden this paper, we shall not discuss the application to elliptic boundary value problems here. This will be postponed to a later paper [6]. Instead, we present the basics of the new approach. The first $\mathcal{H}$-matrix format described in Section 2 is the simplest example. Because of its simple structure, one can explicitly count the work of operations like matrix-matrix multiplication or matrix inversion (cf. Section 3). In spite of the simple structure, we show in Section 4 that these $\mathcal{H}$-matrices allow an exact inversion of tridiagonal matrices.

In order to define the $\mathcal{H}$-matrices, we introduce in Subsection 2.1 the $\mathcal{H}$-partitioning of the index set I. This corresponds to a variable vector block-partitioning. After defining a non-standard matrix-partitioning (partitioning of $I \times I$), we present the definition of an $\mathcal{H}$-matrix in Subsection 2.3.

A more realistic example with respect to interesting applications is the second example of $\mathcal{H}$-matrices introduced in Section 5. With the techniques exercised in Section 3, one finds that the matrix operations have the same order of complexity as for the first example. However, now it is possible to approximate matrices corresponding to integral operators as they typically arise in boundary element methods.

In a sequential paper [6] we shall describe the generalisations to two– and three-dimensional boundary value problems and integral operators. Then the block partitioning which is fixed in the present first and second examples of Sections 2 and 5 must be replaced by an $\mathcal{H}$-partitioning of $I \times I$ and a criterion for the selection must be given.

2 Introductory Example

We start with the simplest example of $\mathcal{H}$-matrices. The hierarchy ("$\mathcal{H}$") is firstly a hierarchy of block partitionings. Therefore, we have first to discuss the block partitionings.

2.1 The Vector Case

2.1.1 Partitioning of a Vector

We consider a vector space of vectors $a = (a_i)_{i \in I}$, where I is a finite index set (e.g., $I = \{1, ..., n\}$). The usual block partitioning consists of a (fixed) partitioning of I into disjoint subsets, i.e., $P = \{I_j : 1 \le j \le k\}$ with

$$I = \dot{\bigcup_{j=1}^{k}} I_j. \tag{1}$$

The jth block of a vector a is $(a_i)_{i \in I_j}$.

In the following we want to control the granularity of the block partitioning, i.e., instead of a fixed partitioning we need a family containing coarse partitionings as well as fine ones. For this purpose, we introduce a set of *hierarchical* partitionings which we call "$\mathcal{H}$-partitionings".

2.1.2 $\mathcal{H}$-Tree over I

In the sequel, we use a tree structure. If t is a vertex of the tree T, $S(t)$ denotes the *set of sons* of t. A vertex $t \in T$ is a *leaf* of the tree, if $S(t) = \emptyset$. $\mathcal{L}(T)$ is the *set of leaves* of T. The root is the unique vertex without parent element. We summarise:

$$\begin{aligned} S(t) &:= \{s \in T : s \text{ is son of } t\} \qquad \text{for } t \in T, \\ \mathcal{L}(T) &:= \{t \in T : S(t) = \emptyset\}. \end{aligned}$$

In the next definition, property (i) is mentioned for the sake of readability although it can be omitted since it follows from (iv).

[2]Do not confound $\mathcal{H}$-matrices with H-matrices (cf. Definition 6.6.7 in Hackbusch [4]).

Definition 2.1 *Let I be an index set. A tree T is called an $\mathcal{H}$-tree (based on I) if the following conditions hold:*

(i) All vertices $t \in T$ are subsets of I.

(ii) $I \in T$.

(iii) $\#S(t) \neq 1$ for all $t \in T$.

(iv) If $t \in T$ is no leaf, $S(t)$ contains disjoint subsets of I and t is the union of its sons, i.e.,

$$t = \dot{\bigcup_{s \in S(t)}} s. \tag{2}$$

We conclude that I is the root of T.

Example 2.2 *Let $I = \{1, ..., n\}$ and $n = 2^p$. The partitioning of level p (finest partitioning) consists of n one-element subsets,*

$$I_1^p = \{1\},\ I_2^p = \{2\}, \ldots,\ I_n^p = \{n\}.$$

On level $p-1$, two subsets from level p are combined. These subsets are

$$I_1^{p-1} = \{1,2\},\ I_2^{p-1} = \{3,4\}, \ldots,\ I_{n/2}^{p-1} = \{n-1, n\}.$$

Similarly, we obtain 4-element subsets of level $p-2$, etc. Finally, at level 0, the whole index set $I_1^0 = I = \{1, 2, \ldots, n\}$ is the only block. This defines a binary tree T with the vertices $\{I_i^\ell : 0 \le \ell \le p,\ 1 \le i \le 2^\ell\}$. I is the root. The vertices at level p are the leaves. The sons of I_i^ℓ ($\ell < p$) are $I_{2i-1}^{\ell+1}$ and $I_{2i}^{\ell+1}$.

We use the notation $t \to s$ to express that $s \in S(t)$. Then the pair (t, s) is an (directed) edge in the tree (graph) T. If there is a path $t = t_0 \to t_1 \to \ldots \to t_k = s$ from t to s (including the case $t = s$, i.e., $k = 0$), we write

$$s \in S^*(t).$$

The characteristic properties of the $\mathcal{H}$-tree T are described in

Remark 2.3 *(a) Let $s, t \in T$ with $s \neq t$. Then exactly one of the following three cases holds:*

(i) $s \subset t$. Then $s \in S^*(t) \backslash \{t\}$.

(ii) $t \subset s$. Then $t \in S^*(s) \backslash \{s\}$.

(iii) $s \cap t = \emptyset$. Then there is a unique smallest[3] $r \in T$ with $s, t \in S^*(r)$.

(b) For any $t \in T$, $S^*(t)$ is a subtree[4] of T satisfying (i), (iii), (iv) from Definition 2.1.

(c) For any $t \in T$,

$$t = \dot{\bigcup_{s \in \mathcal{L}(S^*(t))}} s. \tag{3}$$

Proof. (c) Introduce $S_\ell(t) := \{s \in S^*(t) : \text{there is a path } t \to \ldots \to s \text{ of length} \le \ell\}$. Use induction to prove $t = \dot\bigcup_{s \in S_\ell(t)} s$ for $\ell = 0, 1, \ldots$ and $S_\ell(t) = \mathcal{L}(S^*(t))$ for sufficiently large ℓ. ■

2.1.3 *T*-Partitioning

In the following, we restrict all block partitionings to those which are built by blocks contained in the tree T. For these ones we use the name T-block partitioning.

Definition 2.4 *A block partitioning $P = \{I_j : 1 \le j \le k\}$ of I is called a T-block partitioning (or briefly, a T-partitioning) if $P \subset T$ holds besides (1).*

The set of all such T-partitionings is denoted by $\mathcal{P}(T)$. The next remark shows that T-partitionings can uniquely be described by means of $\mathcal{H}$-subtrees (these are subtrees of T with the $\mathcal{H}$-tree property defined in Definition 2.1).

Remark 2.5 *(i) $\mathcal{P}(T) = \{\mathcal{L}(T') :\ T'$ subtree of T and $\mathcal{H}$-tree$\}$.*

(ii) There is a one-to-one mapping between T-partitionings and $\mathcal{H}$-subtrees T' given by $T' \longmapsto P := \mathcal{L}(T') \in \mathcal{P}(T)$.

Proof. Part (i) follows from (ii). For the proof of (ii) let an $\mathcal{H}$-subtree T' be given. Then (2) ensures $I = \dot\bigcup_{s \in \mathcal{L}(T')} s$; hence, $P := \mathcal{L}(T')$ is a T-partitioning. If a T-partitioning $P = \{I_j : 1 \le j \le k\}$ is given, consider the subtree T' of T consisting of all $t \in T$ with $t \cap I_j = I_j$ or $t \cap I_j = \emptyset$ for all $I_j \in P$. ■

[3] Here, "smallest r" means that there is no proper subset $r' \subset r$, $r' \in T$, with $s, t \in S^*(r')$.

[4] T' is called a *subtree* of T, if the vertices of T' form a subset of those of T and all edges of T' belong to T.

2.1.4 Ordered Index Set

Assume that the index set is ordered. Without loss of generality we may assume $I = \{1, \dots, n\}$. Often, one requires that the subsets $t \in T$ of I consist of consecutive indices, i.e., there are first and last elements $n_0(t), n_1(t) \in I$ such that

$$t = \{i \in I : n_0(t) \le i \le n_1(t)\} \qquad \text{for all } t \in T. \tag{4}$$

The blocks described in Example 2.2 are of this type.

2.2 The Matrix Case

2.2.1 Tensor Block Partitionings versus General Partitionings

Given a block partitioning P of I, the traditional block partitioning of a matrix is given by the product

$$P_2 := P \times P = \{I' \times I'' : I', I'' \in P\}.$$

This means that the subblocks of a matrix A are $A^{ij} = (a_{\alpha\beta})_{\alpha \in I_i, \beta \in I_j}$, where $I_i, I_j \in P$. Hence, row- and column-wise the same blocks I_i and I_j are used. This implies that the size of the blocks cannot be a function of $i - j$ (distance from the diagonal). In later applications we need finer blocks close to the diagonal and coarser ones far away (compare the panel clustering technique in [7]).

We obtain a richer structure, when we look for general block partitionings P_2 of $I \times I$ (not only for tensor product blocks). A *general block partitioning* of $I \times I$ is an partitioning of $I \times I$, where we allow general subsets of $I \times I$. Here, we restrict ourselves to a smaller set of general partitionings ($\mathcal{H}$-partitionings) which are hierarchically structured as in the vector case, but now the $\mathcal{H}$-partitionings are based on the index set $I \times I$ instead of I. Details will be postponed to the subsequent paper [6], since for the cases considered here, we are able to define the partitionings explicitly. In particular, we will describe two partitionings. A very simple one is constructed below, while the partitioning from Section 5 is a prototype for more realistic partitionings needed for boundary value problems.

2.2.2 A Special Non-Tensor Partitioning

Let T be the $\mathcal{H}$-partitioning of I defined in Example 2.2. More generally, we admit binary trees with the property (4). The block partitioning $P_2 = P_2(I, T)$ of $I \times I$ induced by T is defined recursively over the depth of the tree T (largest pathlength in T).

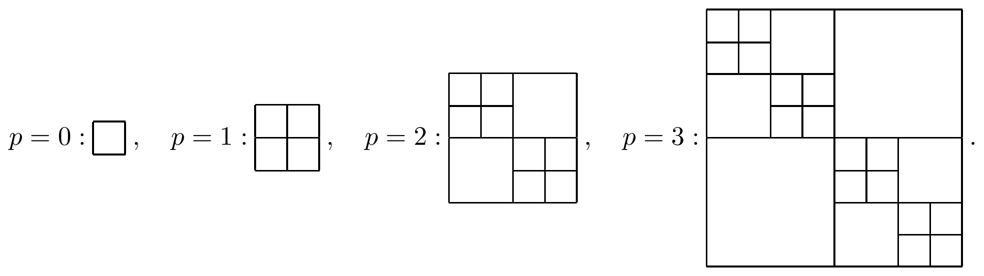
In the trivial case of depth $= 0$ (i.e., $T = \{I\}$), define $P_2(I, T) := \{I \times I\}$.

If depth $= 1$, $I \in T$ has exactly two sons $\{I_1, I_2\}$. Then, $P_2(I, T) := \{I_1 \times I_1, I_1 \times I_2, I_2 \times I_1, I_2 \times I_2\}$ is the standard 2×2-block partitioning of $I \times I$ induced by the vector partitioning $\{I_1, I_2\} = \mathcal{L}(T)$.

If depth > 1, define first the block partitioning $P_2' = \{I_1 \times I_1, I_1 \times I_2, I_2 \times I_1, I_2 \times I_2\}$ discussed above involving the two sons I_1, I_2 of I. Next consider the subtrees $T_k := S^*(I_k)$ for $k = 1, 2$ and replace the blocks $I_k \times I_k$ in P_2' by the finer partitionings $P_2(I_k, T_k)$, i.e.,

$$P_2(I, T) := P_2(I_1, T_1) \cup \{I_1 \times I_2\} \cup \{I_2 \times I_1\} \cup P_2(I_2, T_2). \tag{5}$$

In the case of T from Example 2.2, the tree has the depth p. The corresponding block partitionings $P_2(I, T)$ are depicted below. For $p > 1$, the partitioning is no more of tensor-product form.



(6)

2.3 Hierarchical $\mathcal{H}$-Matrices

In the following, P_2 is a block partitioning of $I \times I$ not restricted to tensor-product partitionings, e.g., P_2 is the partitioning constructed above.

Definition 2.6 *Let P_2 be a block partitioning of $I \times I$ and $k \in \mathbb{N}$. The underlying field of the matrices is $\mathbb{K}$. The set of $\mathcal{H}$-matrices induced by P_2 is*

$$\mathcal{M}_{\mathcal{H},k}(I \times I, P_2) := \{M \in \mathbb{K}^{I \times I} : \text{ each block } M^b,\ b \in P_2,\ \text{satisfies } \operatorname{rank}(M^b) \leq k\}. \tag{7}$$

We call a matrix A an *Rk-matrix* if $\operatorname{rank}(A) \leq k$.

In this section, we restrict our considerations to the simplest case $k = 1$. The resulting *R1*-matrices are discussed in the next subsection.

Another (recursive) description of an $n \times n$ $\mathcal{H}$-matrix A with $n = 2^p$ $(p > 0)$ corresponding to the block partitioning P_2 from §2.2.2 can be given by the requirement that A has the block structure

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \text{ with } \frac{n}{2} \times \frac{n}{2}\ \mathcal{H}\text{-matrices } A_{ii} \text{ and } R1\text{-matrices } A_{12}, A_{21}, \tag{8}$$

where the recursion terminates for 1×1 $\mathcal{H}$-matrices, which are usual 1×1 matrices.

We mentioned in the introduction that the panel clustering method is based on similar ideas. However, the matrix representation of the panel clustering technique corresponds to a different block partitioning of $I \times I$. There the blocks are of the form $\{i\} \times \{t_{j,i}\}$, where $i \in I$ and $\{t_{j,i} : j \in I\}$ is a T-block partitioning of I. This partitioning may be different for each row index i.

2.4 *R1*-Matrices

2.4.1 Properties, Multiplication of *R1*-Matrices

Any $n \times m$-matrix A of rank ≤ 1 (abbreviation: *R1*-matrix or matrix of *R1*-type) can be written in the form

$$A = a * b^H \quad (\text{notation: } A = [a, b]\,) \tag{9}$$

with $a \in \mathbb{K}^n$, $b \in \mathbb{K}^m$ and b^H being the Hermitian transposed of b. Properties of *R1*-matrices[5] are listed in

Remark 2.7 *(a) The amount of storage is $n + m$ (a and b to be stored).*
(b) The amount of work for the matrix-vector multiplication $A * c$ $(c \in \mathbb{K}^m)$ are $2m - 1$ operations to obtain[6] *$\alpha * a$, and $2m + n - 1$ operations, if the multiplication in $\alpha * a$ is performed explicitly.*
(c) Let $A = [a, b]$. Then also $A^H = [b, a]$ is of the form (9).
(d) R1-matrices have a right- and left-ideal property: Multiplication from the right or left by an R1-matrix yields again an R1-matrix. Even if B is a more general matrix, $B * A$ with $A = [a, b]$ requires only the amount of work for computing $B * a$ and yields $B * A = [B * a, b]$. Similarly for $A * B$. If A and B are R1-matrices, $A * B$ or $B * A$ need only one scalar product.
(e) Let $A = [a, b]$. The evaluation of any of the entries $A_{ij} = a_i b_j$ requires exactly one operation.
(f) Let $A = [a, b]$. A complete row $a_i b^H$ [or column $b_j a$] of A requires m [n] operations.

2.4.2 Sums of *R1*-Matrices, Singular-Value Decomposition

Let A and A' be two *R1*-matrices. Then, in general, the sum is not an *R1*-matrix but of rank 2. A suitable approximating *R1*-matrix is the subject of the next considerations.

The singular-value decomposition of an arbitrary $n \times m$-matrix A is

$$A = U * D * V,$$

where U is a unitary $n \times n$-matrix, V a unitary $m \times m$-matrix and D a diagonal $n \times m$ matrix. D contains the "singular values" $d_i \geq 0$. Without loss of generality we may assume $d_1 \geq d_2 \geq \ldots$. Then $k = rank(A)$ is the maximal index k with $d_k > 0$.

[5] R1-matrices are used, since they need very few storage. However, for small blocks this does not pay. Therefore, in practice, one should change definition (7) and use the standard format for 2×2 or even larger blocks.

[6] This suggests using a triple (α, a, b) for $\alpha[a, b]$.

Let A an arbitrary $n \times m$-matrix of the rank k. When we look for an approximate matrix of the rank $k' \in [1, k]$, the matrix

$$A' = U * D' * V \quad \text{with } D' := \operatorname{diag}\{d_1, ..., d_{k'}, 0, ..., 0\}$$

is of rank k' and has the smallest Frobenius norm $\|A - A'\|_F$.

This construction is easily applicable for $k' = 1$ in order to replace the sum $A + B$ of rank 2 by a new approximating *R1*-matrix C (here only eigenvalues and eigenvectors of 2×2-matrices are needed; see proof of Remark 2.8 below). This approximation even allows an *error estimation* by $\|(A + B) - C\|_F$ or $\|(A + B) - C\|_F / \|C\|_F$. The approximate sum C of A and B is a projection of $A + B$ onto the set of *R1*-matrices. We use the notation $C = A +_{R1} B$ or, in the general case of rank-k-matrices,

$$C = A +_{Rk} B \tag{10}$$

for the truncated sum.

For the convenience of the reader, we give the details of the $(+_{R1})$-summation procedure. Given $A = [a_1, b_1]$ and $B = [a_2, b_2]$, let $\Sigma := A+B$ be the true sum and $\Sigma = U*D*V$ its singular value decomposition. The rows of the unitary matrix V are the (normalised) eigenvectors of $\Sigma^H\Sigma$. Since only eigenvectors v corresponding to non-zero eigenvalues are of interest, we may restrict v to the span of $\{b_1, b_2\}$. The ansatz $v = \alpha_1 b_1 + \alpha_2 b_2$ leads to the eigenvalue problem $\lambda\alpha = G_a G_b \alpha$ with $\alpha = (\alpha_1, \alpha_2)^T$ and the 2×2 Gram matrices $G_a = (a_i^H a_j)_{i,j=1,2}$ and $G_b = (b_i^H b_j)_{i,j=1,2}$. Choosing the normalised eigenvector $v = \alpha_1 b_1 + \alpha_2 b_2$ corresponding to the larger of the two eigenvalues λ_1, λ_2, we can represent the truncated sum $C = A +_{Rk} B$ as $[a_3, b_3]$ with $b_3 := v$ and $a_3 := \Sigma v = (Av) + (Bv)$.

Remark 2.8 *The R1-addition $+_{R1}$ of two $n \times m$-matrices costs $9(n + m) + 29$ operations*[7].

Proof. Let $A = [a_1, b_1]$, $B = [a_2, b_2]$ and $C := A +_{R1} B = [a_3, b_3]$. The two Gram matrices[8] $G_a = (a_i^H a_j)_{i,j=1,2}$ and $G_b = (b_i^H b_j)_{i,j=1,2}$ involve six scalar products resulting in $6(n+m)-6$ operations. Let v be the eigenvector of $G_a G_b$ corresponding to the larger eigenvalue. The solving of the 2×2 eigenvalue problem costs $O(1)$ operations. $b_3 := v_1 b_1 + v_2 b_2$ including the normalisation requires $3m + O(1)$ operations. Finally, $a_3 := Ab_3 + Bb_3$ needs $3n + O(1)$ operations exploiting the structure $b_3 := v_1 b_1 + v_2 b_2$ and the already computed values $b_i^H b_j$. ■

Remark 2.9 *If $A = [a, b]$ is an R1-matrix and A' a submatrix, then also A' is an R1-matrix of the form $A' = [a', b']$ with respective subblocks a', b' of a, b.*

2.5 *Rk*-Matrices

If we replace *R1*-matrices by *Rk*-matrices, the following properties hold for fixed k:

- The storage for $n \times m$ *Rk*-matrices is $O(n + m)$.
- The product $A * B$ requires k^2 scalar products.
- The truncation of $A+B$ to an *Rk*-matrix requires the solution of a $2k \times 2k$ eigenvalue problem. However, the additional amount of work is $O(1)$ independent of the dimensions n, m.
- The operation count for the various matrix-vector and matrix-matrix operations described below are of the same order (only the constants are different).

We conclude that there is no problem in using *Rk*-matrices. In the following, we use *R1*-matrices only to simplify the presentation.

[7] A reduction can be achieved by the hint given in Footnote 8.

[8] In order not to recompute the scalar products $a_i^H a_i$ and $b_i^H b_i$, $i = 1, 2$, every time, it is advantageous to compute $a^H a$ and $b^H b$ once for all for any R1-matrix $[a, b]$.

2.6 Uniform $\mathcal{H}$-Matrices

In general, the addition of *Rk*-matrices cannot be performed exactly. In the following, we describe a special situation, where the addition is exact.

So far, an *Rk*-matrix $\sum_{i=1}^{k}[a_i, b_i]$ could be formed with arbitrary vectors a_i, b_i. Another situation occurs if we fix two bases

$$\{a_i : 1 \leq i \leq k\},\ \{b_j : 1 \leq j \leq k\} \subset \mathbb{K}^I \tag{11}$$

and form the *R1*-matrices

$$\{[a_i, b_j] : 1 \leq i, j \leq k\}, \tag{12}$$

which span a subspace $V_k \subset \mathbb{K}^{n\times m}$. We write $V_k = V_k(I \times I)$ for this space of matrices over the index set $I \times I$.

Remark 2.10 *(a) The subspace $V_k = V_k(I \times I)$ spanned by (12) consists of Rk-matrices. Furthermore, $\dim V_k = k^2$ holds.*

(b) If the Rk-matrices A, B belong to V_k, the sum is also an Rk-matrix from V_k.

Let $b = I_1 \times I_2 \subset I \times I$ be some matrix block. The restriction of $[a_i, b_j]$ to b is the block matrix

$$[a_i, b_j]_{|b} = (a_{i,\alpha}\bar{b}_{j,\beta})_{(\alpha,\beta)\in b} \qquad (1 \leq i, j \leq k).$$

Although these block matrices do not necessarily form a basis, they span a subspace denoted by $V_k(b) = V_k(I_1 \times I_2)$.

Definition 2.11 *Let $V_k = V_k(I \times I)$ as before and deduce from V_k the subspaces $V_k(b)$ for all blocks $b \in P_2$. An $\mathcal{H}$-matrix from $\mathcal{M}_{\mathcal{H},k}(I \times I, P_2)$ is a uniform $\mathcal{H}$-matrix, if all block matrices M^b, $b \in P_2$, appearing in (7) belong to $V_k(b)$. The set of uniform $\mathcal{H}$-matrices is denoted by $\mathcal{U}_{\mathcal{H},k}(I \times I, P_2, V_k)$.*

Example 2.12 *Assume $I = \{1, \ldots, n\}$ and a mapping $x : I \to \mathbb{R}$ with $x(\alpha) =: \alpha$ ($\alpha \in I$). Define the vectors $a_i = b_i$ by means of x^{i-1}, i.e., $(a_i)_\alpha = \alpha^{i-1}$ for $\alpha \in I = \{1, \ldots, n\}$. In this case, V_k represents polynomials $\sum_{i,j=0,\ldots,k-1} x^i y^j$. Due to the later Remark 4.2, we may replace the vectors a_i, b_j from above by the scaled ones $a_i' := D_a a_i$ and $b_j' := D_b b_j$, where D_a and D_b are diagonal matrices.*

3 Complexity of the $\mathcal{H}$-Matrix Arithmetic

All statements below correspond to the $\mathcal{H}$-matrix class with P_2 from (5) and $k = 1$. We note that a large part of the operations can be saved if the involved matrices are symmetric and, in particular, if the blocks A_{11}, A_{22} in (8) and their subblocks etc. are identical. In the following, we consider the general case only.

3.1 Storage

As an exercise, we consider the number of blocks in the partitioning $P_2(I, T)$ from (5). Set $N_{block}(p) := \#P_2(I, T)$ for $n = 2^p$. By definition, we have $N_{block}(0) = 1$ and $N_{block}(1) = 4$. Recursion (5) yields $N_{block}(p) = 2 + 2N_{block}(p-1)$ for $p > 1$. This leads to

$$N_{block}(p) = 3n - 2. \tag{13}$$

According to Remark 2.7a, the storage for the *R1*-matrix $[a, b]$ is $N_{R1}(p) := 2n = 2^{p+1}$. For $p = 0$, it is sufficient to store only one real number, i.e., $N_{R1}(0) := 1$. Let $N_{storage}(p)$ be the storage needed for an $\mathcal{H}$-matrix of dimension $2^p \times 2^p$. The recursion (5) yields

$$N_{storage}(p) = 2N_{R1}(p-1) + 2N_{storage}(p-1) = 2^{p+1} + 2N_{storage}(p-1).$$

Together with $N_{storage}(0) = N_{R1}(0) = 1$, we obtain

Lemma 3.1 *The storage requirement for an $n \times n$ $\mathcal{H}$-matrix with $n = 2^p$ is*

$$N_{storage}(p) = (2p + 1)n = (1 + 2\log_2 n)n. \tag{14}$$

3.2 Addition

The sum of two *R1*-matrices is already discussed in Remark 2.8. The costs are denoted by $N_{R1+R1}(p)$.

The exact addition $A+B$ of two $\mathcal{H}$-matrices requires to add all blocks. A^b+B^b, $b\in P_2$. The approximate addition of two $\mathcal{H}$-matrices is defined by replacing the exact operation $+$ by $+_{R1}$ from (10). The result C is denoted by the same symbol: $C=A+_{R1}B$.

Let $n=2^p$ and $p>0$. Denote the cost of the *R1*-addition of two $n\times n$ $\mathcal{H}$-matrices by $N_{H+H}(p)$. Then the recursion $N_{H+H}(p)=2N_{H+H}(p-1)+2N_{R1+R1}(p-1)=2N_{H+H}(p-1)+2(9n+29)$ follows from Remark 2.8. Together with $N_{H+H}(0)=1$ we obtain

$$N_{H+H}(p)=18pn+59n-58. \tag{15}$$

Finally, we discuss the sum $A+B$ of an $n\times n$ $\mathcal{H}$-matrix A and an *R1*-matrix B (notation of the costs: $N_{H+R1}(p)$). Due to Remark 2.9, the *R1*-blocks in $B=\begin{bmatrix}B_{11}&B_{12}\\B_{21}&B_{22}\end{bmatrix}$ can be obtained without arithmetic operations. The recursion $N_{H+R1}(p)=2N_{H+R1}(p-1)+2N_{R1+R1}(p-1)$ and the start $N_{H+R1}(0)=1$ are identical to N_{H+H} and yield $N_{H+R1}(p)=18pn+O(n)$.

Lemma 3.2 *The R1-addition of two $n\times n$ $\mathcal{H}$-matrices or of an $\mathcal{H}$-matrix and an R1-matrix requires $18n\log_2 n+O(n)$ operations.*

Remark 3.3 *The Rk-addition of two uniform $\mathcal{H}$-matrices from $\mathcal{U}_{\mathcal{H},k}(I\times I,P_2,V_k)$ is exact.*

3.3 Matrix-Vector Multiplication

Let A be an $n\times n$ $\mathcal{H}$-matrix and x an n-vector. Decompose A as in (8) and x into the $n/2$-block vectors x_1 and x_2. The multiplication Ax reduces to the computation of $A_{11}x_1$, $A_{12}x_2$, $A_{21}x_1$, $A_{22}x_2$ and their addition. Due to Remark 2.7b, $A_{12}x_2$ and $A_{21}x_1$ cost each $3\frac{n}{2}-1$ operations. Denote the costs of Ax (with $n=2^p$) by $N_{MV}(p)$. The recursion $N_{MV}(p)=2N_{MV}(p-1)+4n-2$ starting with $N_{MV}(0)=1$ yields

$$N_{MV}(p)=4pn-n+2. \tag{16}$$

Lemma 3.4 *The matrix-vector multiplication of an $n\times n$ $\mathcal{H}$-matrix by a (general) vector requires $4n\log_2 n-n+2$ operations.*

The matrix-vector multiplication becomes cheaper if the vector x is sparse. The extreme case is a vector x with only one non-zero component. The proof of the following remark is left to the reader.

Remark 3.5 *Let A be an $n\times n$ $\mathcal{H}$-matrix and x a vector with only m non-zero entries.*
(a) If $m=1$, the multiplication Ax requires $n+\log_2 n$ operations.
(b) For general $m\le n$, the leading term $4n\log_2 n$ from Lemma 3.4 becomes $2(n+m)\log_2 n$.

3.4 Matrix-Matrix Multiplication

Let $n=2^p$. The multiplication of two *R1*-matrices requires $N_{R1*R1}(p):=3n-1$ operations (cf. Remark 2.7d).

Next, we consider the multiplication $A*[a,b]$ $\{$or $[a,b]*A\}$ of an $n\times n$ $\mathcal{H}$-matrix A and the *R1*-matrix $[a,b]$. Since $A*[a,b]=[Aa,b]$, the costs coincide with the operation count of the matrix-vector multiplication from Lemma 3.4: $N_{H*R1}(p):=4n\log_2 n-n+2$. The same number $N_{R1*H}=N_{H*R1}$ holds for $[a,b]*A$.

Let $N_{H*H}(p)$ be the costs for the approximate product $A*_{R1}B$ of the $\mathcal{H}$-matrices. Due to (8), we form the products $A_{11}*_{R1}B_{11}$, $A_{22}*_{R1}B_{22}$ (costs: $2N_{H*H}(p-1)$) and $A_{12}*B_{21}$, $A_{21}*B_{12}$ (costs: $2N_{R1*R1}(p-1)$) as well as $A_{11}*B_{12}$, $A_{22}*B_{21}$, $A_{12}*B_{22}$, $A_{21}*B_{11}$ (costs: $4N_{H*R1}(p-1)$). By Lemma 3.2, the *R1*-additions in $(A_{11}*_{R1}B_{11})+_{R1}(A_{12}*_{R1}B_{21})$, $(A_{21}*_{R1}B_{12})+_{R1}(A_{22}*_{R1}B_{22})$ require $2N_{H+R1}(p-1)$ operations and those in $(A_{11}*B_{12})+_{R1}(A_{12}*B_{11})$, $(A_{22}*B_{21})+_{R1}(A_{21}*B_{11})$ need $2N_{R1+R1}(p-1)$ operations. Altogether, we get the recursion

$$\begin{aligned}N_{H*H}(p)&=2N_{H*H}(p-1)+2N_{R1*R1}(p-1)+4N_{H*R1}(p-1)+2N_{H+R1}(p-1)+2N_{R1+R1}(p-1)\\&=2N_{H*H}(p-1)+26pn+52n-52\end{aligned}$$

with the starting value $N_{H*H}(0)=1$. This leads to $N_{H*H}(p)=13p^2n+65pn-51n+52$.

We summarise:

Lemma 3.6 *The multiplication of two $\mathcal{H}$-matrices requires $13n\log_2^2 n + 65n\log_2 n - 51n + 52$ operations. The multiplication of an $\mathcal{H}$-matrix by an R1-matrix costs $4n\log_2 n - n + 2$, while the multiplication of two R1-matrices needs $3n - 1$ operations.*

3.5 Matrix Inversion

In the following, we assume that an $\mathcal{H}$-matrix A of size $n \times n$ with $n = 2^p$ is given and we try to approximate the inverse A^{-1} by an $\mathcal{H}$-matrix $B = Inv_{R1}(A)$. Again, we use induction with respect to the depth p of block structure. For $p = 0$, $Inv_{R1}(A) := A^{-1}$ is defined as the exact inverse of the 1×1-matrix A. Having defined Inv_{R1} on level $p - 1$, the (exact) inverse of A with block structure (8) is

$$A^{-1} = \begin{bmatrix} A_{11}^{-1} + A_{11}^{-1}A_{12}S^{-1}A_{21}A_{11}^{-1} & -A_{11}^{-1}A_{12}S^{-1} \\ -S^{-1}A_{21}A_{11}^{-1} & S^{-1} \end{bmatrix} \tag{17}$$

with the Schur complement $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$. Since A_{11} is an $\mathcal{H}$-matrix of level $p - 1$, $Inv_{R1}(A_{11})$ is already defined. Because A_{21}, A_{12} are *R1*-matrices, the exact product $A_{21}A_{11}^{-1}A_{12}$ is replaced by the *R1*-matrix $A_{21}Inv_{R1}(A_{11})A_{12}$ (cf. Remark 2.7d). Then $\tilde{S} := A_{22} -_{R1} A_{21}Inv_{R1}(A_{11})A_{12}$ defines the $\mathcal{H}$-matrix approximating S. Next, $Inv_{R1}(\tilde{S})$ can be performed. So far, the computational work amounts to $2N_{inv}(p-1) + N_{H*R1}(p-1) + N_{R1*R1}(p-1) + N_{H+R1}(p-1)$.

The approximation of $-A_{11}^{-1}A_{12}S^{-1}$ by $-Inv_{R1}(A_{11})A_{12}Inv_{R1}(\tilde{S})$ and of the similar block $-S^{-1}A_{21}A_{11}^{-1}$ by $-Inv_{R1}(\tilde{S})A_{21}Inv_{R1}(A_{11})$ costs $3N_{R1*H}(p-1)$ (note that one product, e.g., $A_{11}^{-1}A_{12}$ is already known from the computation of S).

It remains to approximate the left upper block in A^{-1}. Since $S^{-1}A_{21}A_{11}^{-1}$ and $A_{11}^{-1}A_{12}$ are already approximated, $N_{H+R1}(p-1) + N_{R1*R1}(p-1)$ operations complete the computation of $Inv_{R1}(A)$.

The sum of all costs amounts to

$$N_{inv}(p) = 2N_{inv}(p-1) + 4N_{H*R1}(p-1) + 2N_{H+R1}(p-1) + 2N_{R1*R1}(p-1).$$

The previous results yield $N_{inv}(p) = 2N_{inv}(p-1) + 4 * 4(p-1)\frac{n}{2} + 2 * 18(p-1)\frac{n}{2} + O(n) = 2N_{inv}(p-1) + 26pn + 34n - 110$. Together with $N_{inv}(0) = 1$, we obtain $N_{inv}(p) = 13p^2n + 47pn - 109n + 110$.

Lemma 3.7 *The approximate inversion of an $\mathcal{H}$-matrix requires $13n\log_2^2 n + 47n\log_2 n - 109n + 110$ operations.*

3.6 LU-Decomposition

It is of course possible to compute the (approximate) LU-decomposition LU of A with normalised lower triangular $\mathcal{H}$-matrix L and upper triangular $\mathcal{H}$-matrix U. Then the computation of $L^{-1}x$ or $U^{-1}x$ requires $2n\log_2 n + O(n)$ operations. The LU-decomposition needs $6n\log_2^2 n + O(n\log_2 n)$ operations.

4 Properties of the $\mathcal{H}$-Matrices

We have seen that, in general, neither the sum nor the product of two $\mathcal{H}$-matrices from $\mathcal{M}_{\mathcal{H},k}$ are again of the same type so that a "rounding" is necessary. In the following, we list some properties which hold exactly. The first statement is already known.

Remark 4.1 *(a) The matrix-vector multiplication Ax for $A \in \mathcal{M}_{\mathcal{H},k}$ is exact.*
(b) Let $A \in \mathcal{M}_{\mathcal{H},\ell}$ and B an Rk-matrix. Then AB and BA are again Rk-matrices.

Let D be a diagonal matrix. Then the block structure of $A \in \mathcal{M}_{\mathcal{H},k}$ is not changed by a multiplication by D. Furthermore, $DA, AD \in \mathcal{M}_{\mathcal{H},k}$ shows that the product can be performed exactly. This can also be expressed as follows.

Remark 4.2 *$\mathcal{M}_{\mathcal{H},k}$ is invariant with respect to diagonal scaling.*

Although this fact seems trivial, it implies that any kind of equilibration of the matrix entries is unnecessary.

The $\mathcal{H}$-matrix set $\mathcal{M}_{\mathcal{H},1}(I \times I, P_2)$ with P_2 from (5) is very simple. Nevertheless, it is the appropriate format for the treatment of tridiagonal matrices.

Proposition 4.3 *Let $A \in \mathbb{K}^{I\times I}$ be a tridiagonal matrix. Then, A and A^{-1} belong to $\mathcal{M}_{\mathcal{H},1}(I \times I, P_2)$. The matrix $Inv_{R1}(A)$ from §3.5 is the exact inverse A^{-1}.*

Proof. The statement $A, A^{-1} \in \mathcal{M}_{\mathcal{H},1}$ holds for $p = 0$. In the following, we assume the assertion for $p-1$, i.e., $n/2$.

a) By induction, A_{11}, A_{22} from (8) are $\mathcal{H}$-matrices. A_{12} has at most one non-zero entry. Hence A_{12} is an *R1*-matrix $[a, b]$ with $a_i = 0$ except the last index and $b_j = 0$ except the first index. A_{21} has the transposed structure. This proves $A \in \mathcal{M}_{\mathcal{H},1}$ for level p.

b) The proof of $A^{-1} \in \mathcal{M}_{\mathcal{H},1}$ starts with the Schur complement $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$. The characterisation of A_{12} and A_{21} in part a) shows that $A_{21}A_{11}^{-1}A_{12}$ has only one non-zero entry in the (1,1)-position. Therefore, S remains a tridiagonal matrix and by induction $S^{-1} \in \mathcal{M}_{\mathcal{H},1}$ follows. The off-diagonal blocks $-A_{11}^{-1}A_{12}S^{-1}$ and $-S^{-1}A_{21}A_{11}^{-1}$ are of *R1*-type. The first block can be written as the inverse of the Schur complement $S = A_{11} - A_{12}A_{22}^{-1}A_{21}$. ■

Corollary 4.4 *Proposition 4.3 holds also for band matrices with $2k$ off-diagonals with $k > 1$, if $\mathcal{H}$-matrices from $\mathcal{M}_{\mathcal{H},k}(I \times I, P_2)$ are used.*

5 A Second Example for $\mathcal{H}$-Matrices

For many purposes, the class $\mathcal{M}_{\mathcal{H},k}(I \times I, P_2)$ given above is not dense enough around the diagonal. In the following, we present a richer block partitioning P_2' of $I \times I$, which still leads to the same orders of complexity as obtained before.

5.1 The Block Partitioning P_2'

We assume that the $\mathcal{H}$-tree T is the same as before. First, we define $\mathcal{N}$- and $\mathcal{N}^*$-matrices ($\mathcal{N}$ abbreviates "neighbourhood"; block matrices of type $\mathcal{N}$ will be used for neighbouring blocks from T).

Let $n = 2^p$. An $n \times n$-matrix A is an $\mathcal{N}$-matrix (matrix of $\mathcal{N}$-type) if $p = 0$ or if it has the block structure

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \text{ with } \frac{n}{2} \times \frac{n}{2} Rk\text{-matrices } A_{11}, A_{12}, A_{22} \text{ and } \mathcal{N}\text{-matrix } A_{21}. \tag{18}$$

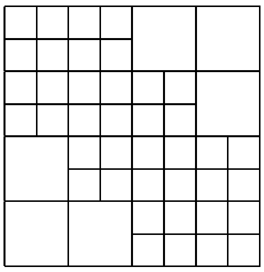
Similarly, we define the transposed type: A is an $\mathcal{N}^*$-matrix if A^T is of $\mathcal{N}$-type, i.e., in (18) A_{11}, A_{21}, A_{22} are Rk-matrices and A_{12} is an $\mathcal{N}^*$-matrix (if $p > 0$). The set of these $\mathcal{N}$- and $\mathcal{N}^*$-matrices is denoted by $\mathcal{M}_{\mathcal{N},k}(I \times I, P_2')$ and $\mathcal{M}_{\mathcal{N}^*,k}(I \times I, P_2')$, respectively (or briefly, $\mathcal{M}_{\mathcal{N},k}, \mathcal{M}_{\mathcal{N}^*,k}$). The product of two matrices of type $\mathcal{N}$ [or both of type $\mathcal{N}^*$] should be truncated into an Rk-matrix.

Then the $\mathcal{H}$-matrices from $\mathcal{M}_{\mathcal{H},k}(I \times I, P_2')$ corresponding to the new block partitioning P_2' are defined in

Definition 5.1 *$A \in \mathcal{M}_{\mathcal{H},k}(I \times I, P_2')$ if either $n = 1$ ($p = 0$) or if*

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \text{ with } A_{11}, A_{22} \in \mathcal{M}_{\mathcal{H},k},\ A_{12} \in \mathcal{M}_{\mathcal{N},k},\ A_{21} \in \mathcal{M}_{\mathcal{N}^*,k}. \tag{19}$$

Note that the 1×1-matrices possess all types; in particular, they are Rk-matrices. For $p \leq 2$, the block partitioning P_2', which is implicitly defined by (19), is the trivial partitioning into 1×1-blocks. Larger blocks appear the first time for $p = 3$:


(20)

5.2 Complexity

Since the arguments are the same as explained in Section 3, we present the results without further explanations. We abbreviate $\log_2 n$ by p.

The number of *blocks* in B_2' is $N_{block} = 9n - 6p - 8$.

The number of *data* to be stored is $N_{storage} = 6pn + O(n)$.

Concerning the *addition*, the different combinations of types must be considered:

$$N_{Rk+Rk}, N_{N+Rk}, N_{N+N} = O(n),\ \ N_{H+Rk}, N_{H+H} = O(pn).$$

Matrix-vector multiplication:

$$N_{Rk*x} = 3n + O(1),\ \ N_{N*x} = 11n + O(1),\ \ N_{H*x} = 11pn + O(n).$$

Matrix-matrix multiplication:

$$N_{Rk*Rk}, N_{N*Rk}, N_{N*N^*} = O(n),\ \ N_{H*Rk}, N_{H*N} = O(pn),\ \ N_{H*H} = O(p^2 n).$$

Inversion:

$$N_{inv} = O(p^2 n).$$

5.3 Approximation of Integral Operators

We will use the newly defined $\mathcal{H}$-matrices to approximate full matrices arising from integral operators as they appear in the boundary element method. For the general structure of the integral operators we refer to [5], [11]. The true background of this application is the fact that the inverse of the discretisation matrix arising from an elliptic boundary value problem has properties quite similar to the discrete integral operator.

Replacing the integration over the surface simply by an integral over $[0,1]$ and choosing the simplest weakly singular kernel $\kappa(z) := \log(z)$, we obtain the example

$$(Au)(x) := \int_0^1 \log(x-y)u(y)dy \qquad \text{for } x \in [0,1].$$

A typical discretisation like the collocation method with piecewise constant elements for the equidistant[9] interval partitioning

$$[x_{i-1}, x_i],\ x_i = ih,\ i = 1,\ldots,n,\ h = 1/n,$$

with the midpoints $x_{i-1/2} = (i - 1/2)h$ of the intervals as collocation points leads to the matrix (discrete operator)

$$A = (a_{ij})_{i,j=1,\ldots,n} \text{ with } a_{ij} = \int_{x_{j-1}}^{x_j} \log(x_{i-1/2} - y)dy. \tag{21}$$

As in the panel clustering method, one can replace the kernel function $\kappa(x,y) = \log(x-y)$ in a certain range of x, y by an approximation $\tilde{\kappa}(x,y)$ of the form

$$\tilde{\kappa}(x,y) = \sum\nolimits_{\iota \in J} X_\iota(x) Y_\iota(y). \tag{22}$$

The simplest choice of such an approximation is Taylor's formula applied with respect to y (then $J = \{0,1,\ldots,k-1\}$, $X_\iota(x) =$ derivatives of $\kappa(x,\cdot)$ evaluated at $y = y^*$ and $Y_\iota(y) = (y-y^*)^\iota$). In this case, one checks that $\kappa(x,y) = \log(x-y)$ leads to the error estimate

$$|\kappa(x,y) - \tilde{\kappa}(x,y)| \le \frac{1}{k} \frac{1}{(|x-y^*| - |y-y^*|)^k} |y-y^*|^k \quad \text{for } |x-y^*| > |y-y^*|. \tag{23}$$

Error estimates of this kind are studied more generally in the panel clustering technique (cf. [5, (9.7.12a,b)]). Other expansions $\tilde{\kappa}(x,y)$ than the Taylor polynomial are studied in [10].

If κ is replaced in (21) by $\tilde{\kappa}$, the integral becomes

$$\tilde{a}_{ij} = \sum_{\iota \in J} X_\iota(x_{i-1/2}) \int_{x_{j-1}}^{x_j} Y_\iota(y)dy. \tag{24}$$

[9]Non-equidistant partitionings work as well since additional factors according to the subinterval lengths are harmless because of Remark 4.2.

Let $b \in B_2'$ be one block and restrict the indices i, j in (24) to b. Then (24) describes a block matrix $\tilde{A}^b$. Obviously, each term of the sum in (24) is an *R1*-matrix $[a, b]$ with $a_i = X_\iota(x_{i-1/2})$ and $b_j = \int_{x_{j-1}}^{x_j} Y_\iota(y)dy$. Since $\#J = k$, the block $\tilde{A}^b$ is of *Rk*-type.

The first 2×2-block of type *Rk* in (20) corresponds to $0 \le x \le 1/4$ and $1/2 \le y \le 3/4$. Choosing $y^* := 5/8$, we obtain $|x - y^*| \ge 3/8$, $|y - y^*| \le 1/8$ and therefore

$$|y - y^*| \le \eta |x - y^*| \tag{25}$$

with $\eta = 1/3$. One checks that (25) with $\eta = 1/3$ holds for all *Rk*-blocks in the $\mathcal{H}$-matrix. The combination of (23) and (25) yields

$$|\kappa(x, y) - \tilde{\kappa}(x, y)| \le \frac{1}{k} (\frac{\eta}{1 - \eta})^k. \tag{26}$$

Hence, the difference $|a_{ij} - \tilde{a}_{ij}|$ is bounded by $\frac{h}{k}(\frac{\eta}{1-\eta})^k$. In the special case of $\eta = 1/3$, we have $|a_{ij} - \tilde{a}_{ij}| \le \frac{h}{k} 2^{-k}$. The maximum norm error satisfies $\|A - \tilde{A}\|_\infty \le 2^{-k}/k$, where k corresponds to the choice of the *Rk*-matrices.

We summarise:

Proposition 5.2 *Approximate the collocation matrix A from (21) by (22)-(24). Then the resulting approximation $\tilde{A}$ is an $\mathcal{H}$-matrix belonging to $\mathcal{M}_{\mathcal{H},k}(I \times I, B_2')$ from Definition 5.1 and satisfies the error estimate $\|A - \tilde{A}\|_\infty \le 2^{-k}/k$.*

References

[1] S. A. Goreinov, E. E. Tyrtyshnikov, and A. Y. Yeremin: *Matrix-free iterative solution strategies for large dense linear systems.* Numerical Linear Algebra with Applications **4** (1997) 273–294.

[2] W. Hackbusch: *The panel clustering algorithm.* MAFELAP 1990 (J. R. Whiteman, ed.). Academic Press, London, 1990 (Uxbridge, April 1990) pp. 339–348.

[3] W. Hackbusch: *The solution of large systems of BEM equations by the multi-grid and panel clustering technique.* Numerical Methods. Rend. Sem. Mat. Univers. Politecn. Torino. Libreria Editrice Universitaria Levrotto & Bella, Torino, 1991 (Turin, June 1990) pp. 163–187.

[4] W. Hackbusch: *Iterative Solution of Large Sparse Systems.* Springer-Verlag, New York, 1994.

[5] W. Hackbusch: *Integral Equations. Theory and Numerical Treatment.* ISNM 128. Birkhäuser, Basel, 1995.

[6] W. Hackbusch and B.N. Khoromskij: *A Sparse $\mathcal{H}$-Matrix Arithmetic. Part II: Application to Multi-Dimensional Problems.* Computing **64** (2000) 21-47

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

[8] W. Hackbusch and S. A. Sauter: *On the efficient use of the Galerkin method to solve Fredholm integral equations.* Applications of Mathematics **38** (1993) 301–322.

[9] C. Lage: *Softwareentwicklung zur Randelementmethode: Analyse und Entwurf effizienter Techniken.* Dissertation, Universität Kiel 1996.

[10] C. Lage: *Fast evaluation of singular kernel functions by cluster methods.* In preparation (1998).

[11] S. A. Sauter: *Über die effiziente Verwendung des Galerkin-Verfahrens zur Lösung Fredholmscher Integralgleichungen.* Dissertation, Universität Kiel 1992.

[12] E. E. Tyrtyshnikov: *Mosaic-skeleton approximations.* Calcolo **33** (1996) 47–57.