Proceedings of EQUADIFF 10, Prague, August 27–31, 2001

AN INTRODUCTION TO HIERARCHICAL MATRICES

Wolfgang Hackbusch, Leipzig, Lars Grasedyck, Kiel,
Steffen Börm, Leipzig

Abstract. We give a short introduction to a method for the data-sparse approximation of matrices resulting from the discretisation of non-local operators occurring in boundary integral methods or as the inverses of partial differential operators.

The result of the approximation will be the so-called hierarchical matrices (or short $\mathcal{H}$-matrices). These matrices form a subset of the set of all matrices and have a data-sparse representation. The essential operations for these matrices (matrix-vector and matrix-matrix multiplication, addition and inversion) can be performed in, up to logarithmic factors, optimal complexity.

Keywords: hierarchical matrices, data-sparse approximations, formatted matrix operations, fast solvers

MSC 2000: 65F05, 65F30, 65F50, 65N50

1. Introduction

1.1. Overview. $\mathcal{H}$-matrices are based on two observations:

- Integral operators can be efficiently treated by using separable expansions of the corresponding kernel functions (cf. [5] or [10]).
- The inverse of an elliptic partial differential operator can be cast in the form of an integral operator by using Green's functions.

In the first half of this introduction, we will present the $\mathcal{H}$-matrix representation of integral operators using a variant of the panel clustering approach (cf. [5]). The second half is devoted to the application of these techniques to the computation of the inverses of matrices arising in finite element discretisations.

1.2. Model problem: Integral equation. Let us consider an integral operator of the form

$$\mathcal{L}\colon V \to W, \quad u \mapsto \left(x \mapsto \int_\Omega g(x,y)u(y)\,\mathrm{d}y\right), \tag{1}$$

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

$$g\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

which is assumed to be *asymptotically smooth*, i.e., to satisfy

$$(2)\quad |\partial_x^\alpha \partial_y^\beta g(x,y)| \leqslant C_{\mathrm{as1}}(C_{\mathrm{as2}}\|x-y\|)^{-|\alpha|-|\beta|}|g(x,y)|, \quad C_{\mathrm{as1}}, C_{\mathrm{as2}} \in \mathbb{R},\ \alpha,\beta \in \mathbb{N}_0^d.$$

In typical applications, g is non-local, so, contrary to the treatment of differential operators, the finite element discretisation of the operator $\mathcal{L}$ does not lead to a sparse matrix. Due to the lack of sparsity, operations on the discrete matrix are prohibitively expensive.

In this paper, we will focus on the method of $\mathcal{H}$-matrices, a combination of the panel clustering method [5] and the mosaic skeleton matrix approach [10]. This method can deal with, in comparison to the two alternatives mentioned above, relatively general domains and operators.

1.3. Elliptic partial differential equations. Since the inverses of elliptic partial differential operators can be represented by Green's associated function in the form of an integral operator, our approximation scheme extends to the inverses of finite element discretisations of such operators. The kernel function (Green's function) is only necessary for theoretical considerations; in practice one starts with the sparse discretisation of an elliptic partial differential operator and calculates an approximate inverse as in Subsection 4.4.

2. Construction of the cluster tree and block partition

While wavelet techniques can be employed to deal directly with problems in a continuum, $\mathcal{H}$-matrix techniques require a discrete subspace together with the finite element or boundary element basis $(\varphi_i)_{i\in I}$. The corresponding Ritz-Galerkin matrix L is given by

$$(3)\qquad L_{ij} = \langle \varphi_i, \mathcal{L}\varphi_j\rangle_{L^2}.$$

2.1. Cluster tree. Let $\mathcal{T}_I$ be a tree and denote by T_I the set of its nodes. $\mathcal{T}_I$ is called a *cluster tree* corresponding to an index set I, if the following conditions hold:

1. $T_I \subseteq \mathcal{P}(I) \setminus \{\emptyset\}$, i.e., each node of $\mathcal{T}_I$ is a subset of the index set I.
2. I is the root of $\mathcal{T}_I$.
3. If $\tau \in T_I$ is a leaf, then[1] $|\tau| \leqslant C_{\mathrm{leaf}}$, i.e., the leaves consist of a relatively small number of indices.
4. If $\tau \in T_I$ is *not* a leaf, then it has two sons and is their disjoint union.

[1] $|\tau|$ denotes the number of elements in the set τ.

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

The restriction of $\mathcal{T}_I$ to binary trees serves only the purpose of simplifying the presentation of some steps of the algorithms. The extension to more general trees is straightforward.

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

$$\Omega_\tau := \bigcup_{i \in \tau} \Omega_i, \quad \text{where } \Omega_i := \operatorname{supp} \varphi_i \ \text{ for all } \ i \in I.$$

E x a m p l e 2.1 (Construction of cluster trees). A simple method of building a cluster tree is based on geometry-based splittings of the index set. We associate each degree of freedom $i \in I$ with a suitable point $x_i \in \mathbb{R}^d$, e.g., the centre of the support of the corresponding basis function or the corresponding Lagrange point, if Lagrangian finite elements are used.

Let $\{e_1, \ldots, e_d\}$ be an orthonormal basis of $\mathbb{R}^d$, e.g., the basis $\{e_x, e_y, e_z\}$ of the canonical unit vectors in 3D. The following algorithm will split a given cluster $\tau \subseteq I$ into two sons:

```
procedure Split($\tau$);
begin
  { Choose a direction for geometrical splitting of the cluster $\tau$ }
  for $j := 1$ to $d$ do
  begin
    $\alpha_j := \min\{\langle e_j, x_i \rangle :\ i \in \tau\}$;
    $\beta_j := \max\{\langle e_j, x_i \rangle :\ i \in \tau\}$
  end;
  $j_{\max} := \operatorname{argmax}\{\beta_j - \alpha_j :\ j \in \{1, \ldots, d\}\}$;
  { Split the cluster $\tau$ in the chosen direction }
  $\gamma := (\alpha_{j_{\max}} + \beta_{j_{\max}})/2$;
  $\tau_1 := \emptyset$; $\tau_2 := \emptyset$;
  for $i \in \tau$ do
    if $\langle e_{j_{\max}}, x_i \rangle \leqslant \gamma$ then
      $\tau_1 := \tau_1 \cup \{i\}$
    else
      $\tau_2 := \tau_2 \cup \{i\}$;
end
```

2.2. Admissibility condition. Next, we need an *admissibility condition* that allows us to select pairs $(\tau, \sigma) \in T_I \times T_I$ such that the kernel $g(\cdot, \cdot)$ is smooth enough on the domain associated with $\Omega_\tau \times \Omega_\sigma$.

If we assume asymptotically smooth kernels, this requirement will lead to an admissibility condition of the form

$$\min\{\operatorname{diam}(\Omega_\tau), \operatorname{diam}(\Omega_\sigma)\} \leqslant \eta \operatorname{dist}(\Omega_\tau, \Omega_\sigma),$$

where $\eta \in \mathbb{R}_{>0}$ is a parameter controlling the trade-off between the number of admissible blocks, i.e., the algorithmic complexity, and the speed of convergence, i.e., the quality of the approximation.

In typical applications for unstructured grids, the computation of the diameter of a cluster and especially of the distance of two clusters will be too complicated or too time-consuming, so the "minimal" condition (4) will be replaced by a stronger variant, for example by using super-sets of Ω_τ and Ω_σ that are of a simpler structure.

Example (Admissibility by Bounding Boxes). A relatively general and practical admissibility condition for clusters in $\mathbb{R}^d$ can be defined by using *bounding boxes*: We define the canonical coordinate maps

$$\pi_k \colon \mathbb{R}^d \to \mathbb{R}, \quad x \mapsto x_k,$$

for all $k \in \{1, \dots, d\}$. The bounding box for a cluster $\tau \in T_I$ is then given by

$$Q_\tau := \prod_{k=1}^{d} [a_{\tau,k}, b_{\tau,k}], \quad \text{where} \quad a_{\tau,k} := \min(\pi_k \Omega_\tau) \quad \text{and} \quad b_{\tau,k} := \max(\pi_k \Omega_\tau).$$

Obviously, we have $\Omega_\tau \subseteq Q_\tau$, so we can define the admissibility condition

$$\min\{\operatorname{diam}(Q_\tau), \operatorname{diam}(Q_\sigma)\} \leqslant \eta \operatorname{dist}(Q_\tau, Q_\sigma) \tag{2.1}$$

which obviously implies (4). We can compute the diameters and distance of the boxes by

$$\operatorname{diam}(Q_\tau) = \left(\sum_{k=1}^{d} (b_{\tau,k} - a_{\tau,k})^2 \right)^{1/2}$$

and

$$\operatorname{dist}(Q_\tau, Q_\sigma) = \left(\sum_{k=1}^{d} (\max(0, a_{\tau,k} - b_{\sigma,k}))^2 + (\max(0, a_{\sigma,k} - b_{\tau,k}))^2 \right)^{1/2}.$$

2.3. Block tree. The cluster tree can be used to define a *block tree* by forming pairs of clusters recursively:

The block tree corresponding to a cluster tree $\mathcal{T}_I$ and an admissibility condition is constructed by the following procedure:

```
procedure BuildBlockTree($\tau \times \sigma$);
begin
  if $\tau \times \sigma$ is not admissible and $|\tau| > C_{\text{leaf}}$ and $|\sigma| > C_{\text{leaf}}$ then
  begin
    $S(\tau \times \sigma) := \{\tau' \times \sigma' \colon \tau' \in S(\tau), \sigma' \in S(\sigma)\}$;
    for $\tau' \times \sigma' \in S(\tau \times \sigma)$ do
      BuildBlockTree($\tau' \times \sigma'$)
  end
  else
    $S(\tau \times \sigma) := \emptyset$
end
```

By calling this procedure with $\tau = \sigma = I$, we create a block cluster tree with root $I \times I$. The leaves of the block cluster tree form a partition of $I \times I$.

The complexity of algorithms for the creation of suitable cluster trees and block partitions has been analysed in detail in [3]: For typical quasi-uniform grids, a "good" cluster tree can be created in $\mathcal{O}(n \log n)$ operations, the computation of the block partition can be accomplished in $\mathcal{O}(n)$ operations.

3. **R**k-MATRICES

The basic building blocks for $\mathcal{H}$-matrices (defined in Section 4) are **R**k-matrices which are a straightforward representation of low rank matrices. These matrices form subblocks of the $\mathcal{H}$-matrix corresponding to subsets $\tau \times \sigma \subset I \times I$.

Definition 3.1 (**R**k-matrix). A matrix of the form

$$R = AB^T, \qquad A \in \mathbb{R}^{\tau \times k}, B \in \mathbb{R}^{\sigma \times k}$$

is called an **R**k-matrix.

Any matrix of rank at most k can be represented as an **R**k-matrix and each **R**k-matrix has at most rank k. **R**k-matrices have some nice properties, e.g., only $k(n+m)$ numbers are needed to store an **R**k-matrix.

3.1. Discretisation. In the $\mathcal{H}$-matrix representation of matrices, **R**k-matrices will occur only as a representation of *admissible* blocks.

If $\mathcal{L}$ is a differential operator, we have $\operatorname{supp}(\mathcal{L}\varphi_j) \subseteq \operatorname{supp} \varphi_j$, so the matrix blocks corresponding to admissible pairs of clusters are zero.

The situation is more complicated if $\mathcal{L}$ is an integral operator of the type (1): Let $\tau \times \sigma$ be an admissible pair of clusters. Without loss of generality, we may assume that $\operatorname{diam}(\Omega_\tau) \leqslant \operatorname{diam}(\Omega_\sigma)$.

In order to construct a rank k approximation of the block $\tau \times \sigma$, we use an m-th order interpolation scheme[2] with interpolation points $(x_j^\tau)_{j=1}^k$ and the corresponding Lagrange polynomials $(p_j^\tau)_{j=1}^k$ and approximate the original kernel function $g(\cdot,\cdot)$ by its interpolant

$$\tilde{g}(x,y) := \sum_{\iota=1}^{k} p_\iota^\tau(x) g(x_\iota^\tau, y). \tag{6}$$

Combining the asymptotical smoothness assumption (2) with standard interpolation error estimates, we get

$$|g(x,y) - \tilde{g}(x,y)| \leqslant C\left(C_{\mathrm{int}} C_{\mathrm{as2}} \frac{\operatorname{diam}(\Omega_\tau)}{\operatorname{dist}(\Omega_\tau, \Omega_\sigma)}\right)^m \|g\|_{\infty, \Omega_\tau \times \Omega_\sigma},$$

which combined with the admissibility condition (4) yields

$$|g(x,y) - \tilde{g}(x,y)| \leqslant C\left(C_{\mathrm{int}} C_{\mathrm{as2}} \eta\right)^m \|g\|_{\infty, \Omega_\tau \times \Omega_\sigma},$$

so if $\eta < 1/(C_{\mathrm{int}} C_{\mathrm{as2}})$, we get e x p o n e n t i a l convergence of the interpolation if we increase the order m.

By replacing $g(\cdot,\cdot)$ by $\tilde{g}(\cdot,\cdot)$ in (3), we find

$$L_{ij} = \sum_{\iota=1}^{k} \int_\Omega p_\iota^\tau(x) \varphi_i(x)\, \mathrm{d}x \int_\Omega g(x_\iota^\tau, y) \varphi_j(y)\, \mathrm{d}y. \tag{7}$$

We define matrices $A \in \mathbb{R}^{\tau \times k}$ and $B \in \mathbb{R}^{\sigma \times k}$ by setting

$$A_{i\iota} := \int_\Omega p_\iota^\tau(x) \varphi_i(x)\, \mathrm{d}x \quad \text{and} \quad B_{j\iota} := \int_\Omega g(x_\iota^\tau, y) \varphi_j(y)\, \mathrm{d}y$$

and rewrite (7) as[3]

$$L|_{\tau \times \sigma} \approx AB^T,$$

so we have approximated $L|_{\tau \times \sigma}$ by an $\mathbf{R}k$-matrix.

[2] A scheme of this type can be easily constructed by extending one-dimensional interpolation techniques to the multi-dimensional case using tensor products. This leads to the relation $k = m^d$ between the rank of a block and the order of the interpolation.

[3] For a vector v and a subset $\tau \subset I$, $v|_\tau$ is the restriction to the vector $(v_j)_{j \in \tau}$, while for a matrix L and subsets $\tau, \sigma \subset I$ the notation $L|_{\tau \times \sigma}$ is used for the block $(L_{ij})_{i \in \tau, j \in \sigma}$.

3.2. Matrix-vector multiplication. The matrix-vector multiplication $x \mapsto y := Rx$ of an $\mathbf{R}k$-matrix $R = AB^T$ with a vector $x \in \mathbb{R}^\sigma$ can be done in two steps:

1. Calculate $z := B^T x \in \mathbb{R}^k$.
2. Calculate $y := Az \in \mathbb{R}^\tau$.

The transposed $R^T = BA^T$ can be treated analogously and the complexity of the matrix-vector multiplication is $\mathcal{O}(k(|\sigma| + |\tau|)$.

3.3. Truncation. The best approximation of an arbitrary matrix $M \in \mathbb{R}^{\tau\times\sigma}$ by an $\mathbf{R}k$-matrix $\widetilde{M} = \widetilde{A}\widetilde{B}^T$ (in the spectral and Frobenius norm) can be computed using the (truncated) singular value decomposition as follows:

1. Calculate a singular value decomposition $M = U\Sigma V^T$ of M.
2. Set $\widetilde{U} := [U_1 \dots U_k]$ (first k columns), $\widetilde{\Sigma} := \operatorname{diag}(\Sigma_{11}, \dots, \Sigma_{kk})$ (first (largest) k singular values), $\widetilde{V} := [V_1 \dots V_k]$ (first k columns).
3. Set $\widetilde{A} := \widetilde{U}\widetilde{\Sigma} \in \mathbb{R}^{\tau\times k}$ and $\widetilde{B} := \widetilde{V} \in \mathbb{R}^{\sigma\times k}$.

We call $\widetilde{M}$ the truncation of M to the set of $\mathbf{R}k$-matrices. The complexity of the truncation is $\mathcal{O}((|\tau| + |\sigma|)^3)$. If the matrix M is an $\mathbf{R}K$-matrix $M = AB^T$ with $K > k$ then the truncation can be computed in $\mathcal{O}(K^2(|\tau| + |\sigma|) + K^3)$ by the following procedure:

1. Calculate a truncated QR-dec. $A = Q_A R_A$ of A, $Q_A \in \mathbb{R}^{\tau\times K}, R_A \in \mathbb{R}^{K\times K}$.
2. Calculate a truncated QR-dec. $B = Q_B R_B$ of B, $Q_B \in \mathbb{R}^{\sigma\times K}, R_B \in \mathbb{R}^{K\times K}$.
3. Calculate a singular value decomposition $R_A R_B^T = U\Sigma V^T$ of $R_A R_B^T$.
4. Set $\widetilde{U} := [U_1 \dots U_k]$ (first k columns), $\widetilde{\Sigma} := \operatorname{diag}(\Sigma_{11}, \dots, \Sigma_{kk})$ (first (largest) k singular values), $\widetilde{V} := [V_1 \dots V_k]$ (first k columns).
5. Set $\widetilde{A} := Q_A\widetilde{U}\widetilde{\Sigma} \in \mathbb{R}^{\tau\times k}$ and $\widetilde{B} := Q_B\widetilde{V} \in \mathbb{R}^{\sigma\times k}$.

3.4. Addition. Let $R_1 = AB^T, R_2 = CD^T$ be $\mathbf{R}k$-matrices. The sum

$$R_1 + R_2 = [A\,C][B\,D]^T$$

is an $\mathbf{R}K$-matrix with $K = 2k$. We define the *formatted addition* $\oplus$ of two $\mathbf{R}k$-matrices as the best approximation (in the spectral and Frobenius norm) of the sum in the set of $\mathbf{R}k$-matrices, which can be computed as in Section 3.3. The *formatted subtraction* $\ominus$ is defined analogously.

3.5. Multiplication. The multiplication of an $\mathbf{R}k$-matrix $R = AB^T$ by an arbitrary matrix M from the left or right yields again an $\mathbf{R}k$-matrix:

$$\begin{aligned} RM &= AB^T M = A(M^T B)^T, \\ MR &= MAB^T = (MA)B^T. \end{aligned}$$

To calculate the product one has to perform the matrix-vector multiplication $M^T B_i$ for the k columns $i = 1, \dots, k$ of B with the transpose of M, or MA_i for the k columns $i = 1, \dots, k$ of A with the matrix M.

4. $\mathcal{H}$-matrices

Based on the cluster (binary) tree $\mathcal{T}_I$ and the block cluster (quad-) tree $\mathcal{T}_{I\times I}$ we define the $\mathcal{H}$-matrix structure.

Definition 4.1 ($\mathcal{H}$-matrix). Let $L \in \mathbb{R}^{I\times I}$ be a matrix and $\mathcal{T}_{I\times I}$ a block cluster tree of $I \times I$ consisting of admissible and non-admissible leaves. Let $k \in \mathbb{N}$. L is called *$\mathcal{H}$-matrix* of blockwise rank k, if for all admissible leaves $\tau \times \sigma \in T_{I\times I}$

$$\operatorname{rank}(L|_{\tau\times\sigma}) \leqslant k,$$

i.e., each admissible subblock of the matrix is an **R**k-matrix while the non-admissible subblocks corresponding to leafs do not have to bear any specific structure.

Remark 4.2. If $\tau \times \sigma$ is a non-admissible leaf of $\mathcal{T}_{I\times I}$, then either $|\tau| \leqslant C_{\text{leaf}}$ or $|\sigma| \leqslant C_{\text{leaf}}$, which means that the rank is bounded by C_{leaf}.

The storage requirements for an $\mathcal{H}$-matrix are $\mathcal{O}(nk\log(n))$ for the one- and two-dimensional block tree in [4] and [6]. The same bound also holds for any $\mathcal{H}$-matrix based on a sparse block tree (see [3]).

4.1. Matrix-vector multiplication. Let $L \in \mathbb{R}^{I\times I}$ be an $\mathcal{H}$-matrix. To compute the matrix-vector product $y := y + Lx$ with $x, y \in \mathbb{R}^I$, we use the following procedure:

```
procedure MVM($L$, $\tau \times \sigma$, $x$, $y$);
begin
  if $S(\tau \times \sigma) \neq \emptyset$ then
    for each $\tau' \times \sigma' \in S(\tau \times \sigma)$ do
      MVM($L$, $\tau' \times \sigma'$, $x$, $y$)
  else
    $y|_\tau := y|_\tau + L|_{\tau\times\sigma} x|_\sigma$; {unstructured or Rk-matrix}
end
```

The starting index sets are $\tau = \sigma = I$.

The complexity for the matrix-vector multiplication is $\mathcal{O}(kn\log(n))$ under moderate assumptions (see [3]) concerning the locality of the supports of the basis functions φ_i.

4.2. Addition. Let $L, L^{(1)}, L^{(2)} \in \mathbb{R}^{I\times I}$ be $\mathcal{H}$-matrices. The sum $L := L^{(1)} + L^{(2)}$ is an $\mathcal{H}$-matrix with blockwise rank $2k$. The *formatted sum* $\widetilde{L} := L^{(1)} \oplus L^{(2)}$ is defined by the formatted addition of the $\mathbf{R}k$-subblocks:

```
procedure Add($\widetilde{L}$, $\tau \times \sigma$, $L^{(1)}$, $L^{(2)}$);
begin
  if $S(\tau \times \sigma) \neq \emptyset$ then
    for each $\tau' \times \sigma' \in S(\tau \times \sigma)$ do
      Add($\widetilde{L}$, $\tau' \times \sigma'$, $L^{(1)}$, $L^{(2)}$);
  else
    $\widetilde{L}|_{\tau\times\sigma} := L^{(1)}|_{\tau\times\sigma} \oplus L^{(2)}|_{\tau\times\sigma}$; {unstructured or $\mathbf{R}k$-matrix}
end
```

4.3. Multiplication. Let $L, L^{(1)}, L^{(2)} \in \mathbb{R}^{I\times I}$ be $\mathcal{H}$-matrices. The matrix $L := L + L^{(1)} \cdot L^{(2)}$ is an $\mathcal{H}$-matrix with blockwise rank $O(k \log(n))$. The *formatted product* $\widetilde{L} := L \oplus L^{(1)} \odot L^{(2)}$ is defined by using the formatted addition in the $\mathbf{R}k$-subblocks. We distinguish three cases:

1. All matrices are subdivided. The multiplication and addition is done in the subblocks.
2. The target matrix is subdivided and (at least) one of the factors is not subdivided. One has to add the product (small or $\mathbf{R}k$-matrix involved) to the target matrix.
3. The target matrix is not subdivided. This case will be treated in a separate procedure *MulAddRk*.

```
procedure MulAdd($\widetilde{L}$, $\tau$, $\zeta$, $\sigma$, $L^{(1)}$, $L^{(2)}$);
begin
  if $S(\tau \times \zeta) \neq \emptyset$ and $S(\zeta \times \tau) \neq \emptyset$ and $S(\tau \times \sigma) \neq \emptyset$ then
    { Case 1: all matrices are subdivided }
    for each $\tau' \in S(\tau), \zeta' \in S(\zeta), \sigma' \in S(\sigma)$ do
      MulAdd($\widetilde{L}$, $\tau'$, $\zeta'$, $\sigma'$, $L^{(1)}$, $L^{(2)}$);
  else begin
    if $S(\tau \times \sigma) \neq \emptyset$ then
    begin
      {Case 2: the target matrix is subdivided}
      Calculate the product $L' := L^{(1)}|_{\tau\times\zeta} L^{(2)}|_{\zeta\times\sigma}$ (unstructured or $\mathbf{R}k$-matrix)
      and add $L'$ to $\widetilde{L}|_{\tau\times\sigma}$ {formatted addition in subblocks of $\tau \times \sigma$}
    end
    else begin
      {Case 3: the target matrix is not subdivided}
      MulAddRk($\widetilde{L}$, $\tau$, $\zeta$, $\sigma$, $L^{(1)}$, $L^{(2)}$)
      end
    end
  end
```

To cover case 3 we have to multiply two subdivided matrices, truncate the product to the set of **R**k-matrices and add the result to the target matrix. To do this we first calculate the products in the subblocks and truncate them to the set of **R**k-matrices. Afterwards all four **R**k-submatrices are added to the target matrix (extending them by zeros so that all matrices are of the same size) using the formatted addition.

```
  procedure MulAddRk($\widetilde{L}$, $\tau$, $\zeta$, $\sigma$, $L^{(1)}$, $L^{(2)}$);
  begin
    if $S(\tau \times \zeta) = \emptyset$ or $S(\zeta \times \sigma) = \emptyset$ then
    begin
      Calculate the product $L' := L^{(1)}|_{\tau\times\zeta} L^{(2)}|_{\zeta\times\sigma}$ {unstructured or Rk-matrix}
      and add $L'$ to $\widetilde{L}|_{\tau\times\sigma}$ {formatted addition}
    end
    else begin
      for each $\tau' \in S(\tau), \sigma' \in S(\sigma)$ do
      begin
        Initialise $L'_{\tau',\sigma'} := 0$;
        for each $\zeta' \in S(\zeta)$ do
          MulAddRk($L'_{\tau',\sigma'}$, $\tau'$, $\zeta'$, $\sigma'$, $L^{(1)}$, $L^{(2)}$);
        {$L'_{\tau',\sigma'}$ is smaller than $L$ and extended by zeros}
      end;
      $\widetilde{L} := L \oplus \sum_{\tau' \in S(\tau)} \sum_{\sigma' \in S(\sigma)} L'_{\tau',\sigma'}$
    end
  end
```

4.4. Inversion. The inverse of a 2×2 block-matrix can be computed by use of the Schur complement (see [4]). The exact sums and products are replaced by the formatted operations $\oplus, \odot$ and recursively one can define the *formatted inverse* F of L.

```
  procedure Invert($F$, $\tau$, $\sigma$, $L$);
  begin
    if $S(\tau \times \sigma) = \emptyset$ then begin
      Calculate the inverse $F := L^{-1}$ exactly { unstructured small matrix }
    end
    else begin
      $\left\{ S(\tau) = \{\tau_1, \tau_2\},\ S(\sigma) = \{\sigma_1, \sigma_2\},\ F = \begin{bmatrix} F_{11} F_{12} \\ F_{21} F_{22} \end{bmatrix}\ L = \begin{bmatrix} L_{11} L_{12} \\ L_{21} L_{22} \end{bmatrix} \right\}$
    Invert(Y,$\tau_1$, $\sigma_1$, $L_{11}$);
    $S := L_{22} \ominus (L_{21} \odot (Y \odot L_{12}))$;
    Invert($F_{22}$, $\tau_2$, $\sigma_2$, $S$);
    $F_{11} := Y \oplus (Y \odot (L_{12} \odot (F_{22} \odot (L_{21} \odot Y))))$;
    $F_{12} := -Y \odot (L_{12} \odot F_{22})$;
    $F_{21} := -F_{22} \odot (L_{21} \odot Y)$
  end
end
```

5. Numerical examples

In order to demonstrate the advantage of the $\mathcal{H}$-matrix approach, we consider the simple example of the discretisation of the single layer potential on the unit circle in two space dimensions using a Galerkin method with piecewise constant basis functions.

The logarithmic kernel function will be approximated by the interpolation approach given in (6) using tensor product Chebyshev points and corresponding Lagrange polynomials.

We report the relative error $\|L - \widetilde{L}\|_2 / \|L\|_2$ in Table 1.[4] The first column contains the number of degrees of freedom for each discretisation level, the following columns give the relative error. We observe that the error is bounded independently of the discretisation level and that it decreases very quickly when the interpolation order is increased.

n	1	2	3	4	5
1024	0.0357	0.002159	0.0002504	$7.877e-06$	$2.667e-06$
2048	0.03581	0.002185	0.0002507	$7.86e-06$	$2.691e-06$
4096	0.03587	0.002198	0.0002505	$7.865e-06$	$2.68e-06$
8192	0.03589	0.002204	0.0002518	$7.755e-06$	$2.667e-06$
16384	0.03591	0.002207	0.0002526	$7.873e-06$	$2.684e-06$

Table 1. Approximation error for the single layer potential

The time required for matrix vector multiplications is given in Table 2. It was measured on a SUN Enterprise 6000 machine using UltraSPARC II processors running at 248 MHz by taking the time required for 100 matrix vector multiplications and dividing by 100. We can see that the complexity grows almost linearly with respect to the number of degrees of freedom and rather slowly with respect to the interpolation order.

[4] The Euclidean norms are approximated by performing 100 steps of the power iteration.

n	1	2	3	4	5
1024	0.01	0.02	0.01	0.01	0.03
2048	0.02	0.04	0.03	0.05	0.07
4096	0.05	0.11	0.09	0.12	0.17
8192	0.12	0.24	0.19	0.26	0.39
16384	0.27	0.53	0.41	0.56	0.83
32768	0.57	1.15	0.90	1.23	1.90
65536	1.18	2.44	1.96	2.73	4.14
131072	2.45	5.18	4.30	5.89	8.98
262144	5.15	11.32	9.14	12.95	19.78
524288	10.68	23.81	19.62	28.02	43.57

Table 2. Time [sec] required for the matrix vector multiplication

Finally, let us consider the time required for building the $\mathcal{H}$-matrix representation of the discretised integral operator. It is given in Table 3 and was measured on the same machine. The integral of the Lagrange polynomials was computed by using an exact Gauss quadrature formula, while the integral of the kernel function was computed analytically. Once more we observe an almost linear growth of the complexity with respect to the number of degrees of freedom and a slow growth with respect to the interpolation order. Note that even on an old and quite slow processor like the 248 MHz UltraSPARC II, the boundary element matrix for more than half a million degrees of freedom can be approximated with an error less than 0.03% in less than half an hour.

	1	2	3	4	5
1024	0.61	0.93	1.76	3.11	5.60
2048	1.25	2.03	3.85	7.04	12.94
4096	2.56	4.29	8.41	15.82	29.65
8192	5.25	9.16	18.10	35.31	66.27
16384	10.75	19.30	39.32	77.47	146.65
32768	22.15	40.83	85.16	169.16	324.36
65536	45.79	87.32	185.85	368.46	702.63
131072	92.64	180.73	387.63	788.06	1511.66
262144	189.15	378.20	854.75	1775.85	3413.45
524288	388.96	795.84	1743.66	3596.77	6950.55

Table 3. Time [sec] required for building the $\mathcal{H}$-matrix

6. Concluding remarks

Estimates concerning the cost of the matrix operations and the approximation error can be found in [6], [7] and [3]. There are several special variants of the $\mathcal{H}$-matrix technique described in [8]. The treatment of non-quasiuniform finite element meshes is studied in [9]. Even matrix functions like the matrix exponential can be computed effectively: [1], [2].

References

[1] *I. P. Gavrilyuk, W. Hackbusch, B. N. Khoromskij*: $\mathcal{H}$-matrix approximation for the operator exponential with applications. Numer. Math. To appear.

[2] *I. P. Gavrilyuk, W. Hackbusch, B. N. Khoromskij*: $\mathcal{H}$-matrix approximation for elliptic solution operators in cylindric domains. East-West J. Numer. Math. *9* (2001), 25–58.

[3] *L. Grasedyck*: Theorie und Anwendungen Hierarchischer Matrizen. Doctoral thesis, University Kiel, 2001.

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

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

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

[7] *W. Hackbusch, B. N. Khoromskij*: A sparse $\mathcal{H}$-matrix arithmetic: general complexity estimates. J. Comput. Appl. Math. *125* (2000), 479–501.

[8] *W. Hackbusch, B. N. Khoromskij, S. A. Sauter*: On $\mathcal{H}^2$-matrices. Lectures on applied mathematics (Hans-Joachim Bungartz, Ronald H. W. Hoppe, Christoph Zenger, eds.). Springer, Berlin, 2000, pp. 9–29.

[9] *W. Hackbusch, B. N. Khoromskij*: $\mathcal{H}$-matrix approximation on graded meshes. The Mathematics of Finite Elements and Applications X, MAFELAP 1999 (John R. Whiteman, ed.). Elsevier, Amsterdam, 2000, pp. 307–316.

[10] *E. Tyrtyshnikov*: Mosaic-skeleton approximation. Calcolo *33* (1996), 47–57.

Authors' addresses: *Wolfgang Hackbusch*, Max Planck Institute for Mathematics in the Sciences, Inselstrasse 22-26, 04103 Leipzig, Germany, e-mail: `wh@mis.mpg.de`; *Lars Grasedyck*, Mathematisches Seminar Bereich 2, Universität Kiel, Hermann-Rodewald-Strasse 3, 24098 Kiel, Germany; *Steffen Börm*, Max Planck Institute for Mathematics in the Sciences, Inselstrasse 22-26, 04103 Leipzig, Germany, e-mail: `sbo@mis.mpg.de`.