Max-Planck-Institut
für Mathematik
in den Naturwissenschaften
Leipzig

Panel Clustering Techniques and Hierarchical Matrices for BEM and FEM

by

Wolfgang Hackbusch

Preprint no.: 71 2003

Panel Clustering Techniques and Hierarchical Matrices for BEM and FEM*

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

Abstract

The panel clustering method and the hierarchical matrix technique described here are important tools for the efficient treatment of fully populated matrices which arise from boundary element (BEM) problems and elliptic FEM problems.

Keywords. Panel clustering, hierarchical matrices, BEM, fully populated matrices, fast multiplication, efficient matrix operations

1 Introduction

The main background of the so-called 'panel clustering technique' is the efficient numerical treatment of *integral equations*. Therefore, we first remind the reader to the boundary element method and the respective integral equations (see §1.2). The discrete problem is described by a fully populated $n \times n$ matrix. The naive approach requires a storage of the size n^2 and the standard matrix-vector multiplication needs $\mathcal{O}(n^2)$ arithmetical operations. In order to realise the advantages of BEM compared with FEM, it is essential to reduce the order $\mathcal{O}(n^2)$ of the cost to almost $\mathcal{O}(n)$.

The panel clustering technique described in Section 2 allows to reduce the storage and matrix-vector costs from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log^q n)$. The reduction of the memory is in particular important for 3D applications, when $\mathcal{O}(n \log^q n)$ data can easily be stored, while $\mathcal{O}(n^2)$ exceeds the memory bounds. The reduction of the cost for the matrix-vector multiplication is important as well, since this is the essential operation in usual iterative methods for solving the system of linear equations. The essential ingredients of the panel clustering technique are (i) the far field expansion (§2.1) and (ii) the panel cluster tree (§2.2). The section is concluded by hints concerning implementational details (§2.7).

Section 3 presents a second variant of the panel clustering technique. This variant of the panel clustering technique can be generalised to the technique of hierarchical matrices ($\mathcal{H}$-matrices), which is described in Section 4. Again, the $\mathcal{H}$-matrix structure can be used to represent fully populated matrices. This technique allows not only the matrix-vector multiplication, but also matrix operations like matrix-plus-matrix, matrix-times-matrix and even matrix-inversion.

1.1 Notations

We have already used the Landau symbol $\mathcal{O}(f(n))$, which means that the quantity is bounded by $C * f(n)$ as $n \to \infty$ for some positive constant C. For an index set I, the set $\mathbb{R}^I$ denotes the set of (real) vectors $\mathbf{a} = (a_i)_{i \in I}$ indexed by means of I. Similarly, the notation $\mathbb{R}^{I \times J}$ is used for the set of matrices $\mathbf{A} = (a_{i,j})_{i \in I, j \in J}$.

*To appear as Chapter 20 of Vol. I in Erwin Stein, René de Borst, and Thomas J.R. Hughes (eds.): *Encyclopedia of Computational Mechanics*. Wiley, Chichester

item	explanation	reference
$\mathbf{A}, \mathbf{B}, \ldots$	matrices of size $n \times n$	(1.11)
b	block, vertex of T_2	§3.1.1
b_j	BEM basis function	(1.10)
d	spatial dimension of $\mathbb{R}^d$	(1.1)
diam, dist	diameter and distance of clusters	(2.5), (3.1)
I	index set for the matrix entries	§4.1
I_m	index set in the representation of $\tilde{\kappa}$	(2.2)
$J_\tau^\iota, J_\tau^\iota(b_j)$	far field coefficients	§2.4.3
K	integral operator	(1.4)
n	problem dimension, matrix size	(1.10), §4.1
$\mathcal{P}$	set of panels (boundary elements)	§1.2.3
$s(\mathbf{x}, \mathbf{y})$	fundamental solution	§1.2.1
t	triangle (panel) $t \in \mathcal{P}$	§1.2.3
$S, S_2, S_I, S_{I\times I}(\tau)$	set of sons	§2.2, §3.1.1, §4.2
$T, T_2, T_I, T_{I\times I}$	cluster tree, tree of blocks, block cluster tree	§2.2, §3.1.1, §4.2
$\mathbf{u}$	coefficient vector from $\mathbb{R}^n$	(1.10)
V_h	boundary element space	§1.2.3
$\mathbf{x}, \mathbf{y}, \mathbf{z}$	points in $\mathbb{R}^d$	(1.2)
$\mathbf{z}_\tau$	centre of cluster τ	§2.7.2
Γ	surface contained in $\mathbb{R}^d$	§1.2.2
η	parameter in admissibility condition	(2.5), (3.1)
$\kappa(\mathbf{x}, \mathbf{y})$	kernel of integral operator	(1.4)
$\tilde{\kappa}(\mathbf{x}, \mathbf{y}), \kappa_b(\mathbf{x}, \mathbf{y})$	far field approximation of κ	(2.2), (3.3)
ξ, ξ^i	collocation point	§1.2.3
τ (also τ', σ, σ')	cluster, vertex of the tree T	§2.2
$\Phi_\iota, \Phi_\tau^\iota$	expansion functions	(2.2)
$\int_\Gamma \ldots d\Gamma_x$	surface integration	(1.4)
$\#S$	cardinality of the set S, i.e., number of elements	

1.2 The Boundary Element Method (BEM)

1.2.1 The Problem to be Solved

There are several important applications where an elliptic boundary value problem with vanishing source term is to be solved,

$$Lu = 0 \qquad \text{in } \Omega \subset \mathbb{R}^d. \tag{1.1}$$

Here, Ω may be a bounded or unbounded domain. Since L is assumed to have *constant coefficients*, the fundamental solution $s(\mathbf{x}, \mathbf{y})$ is known explicitly. It satisfies $L_x s(\mathbf{x}, \mathbf{y}) = \delta(\mathbf{x} - \mathbf{y})$, where $L_x = L$ is applied to the $\mathbf{x}$-argument and δ is the Dirac function. In the case of $Lu = f \neq 0$, a further integral over Ω appears which can be treated efficiently by means of the hierarchical matrices from Section 4. Examples for L and s are the Laplace problem,

$$L = \Delta, \quad s(\mathbf{x}, \mathbf{y}) = \begin{cases} \frac{1}{2\pi} \log |\mathbf{x} - \mathbf{y}| & \text{for } d = 2 \text{ (i.e., } \mathbf{x}, \mathbf{y} \in \mathbb{R}^2), \\ \frac{1}{4\pi |\mathbf{x} - \mathbf{y}|} & \text{for } d = 3 \text{ (i.e., } \mathbf{x}, \mathbf{y} \in \mathbb{R}^3), \end{cases} \tag{1.2}$$

the Helmholtz problem $L = \Delta + a^2$, $s(\mathbf{x}, \mathbf{y}) = \frac{\exp(ia|\mathbf{x}-\mathbf{y}|)}{4\pi|\mathbf{x}-\mathbf{y}|}$, and the Lamé equation ($d = 3$)

$$\mu \Delta \mathbf{u} + (\lambda + \mu) \nabla \operatorname{div} \mathbf{u} = 0, \tag{1.3}$$

$$\mathbf{S}(\mathbf{x}, \mathbf{y}) = \frac{\lambda + 3\mu}{8\pi (\lambda + 2\mu)} \left\{ \frac{1}{|\mathbf{x} - \mathbf{y}|} \mathbf{I} + \frac{\lambda + \mu}{\lambda + 3\mu} \frac{(\mathbf{x} - \mathbf{y}) (\mathbf{x} - \mathbf{y})^\top}{|\mathbf{x} - \mathbf{y}|^3} \right\}.$$

In the latter example, the fundamental solution $\mathbf{S}(\mathbf{x}, \mathbf{y})$ is matrix-valued. In all examples, $|\mathbf{x} - \mathbf{y}|$ is the standard Euclidean norm of the vector $\mathbf{x} - \mathbf{y} \in \mathbb{R}^d$.

1.2.2 Formulation by an Integral Equation

The advantage of the following integral equation formulation is the fact that the domain of integration is the boundary $\Gamma = \partial\Omega$. Thus the spatial dimension is reduced by one. This advantage is even more essential, if Ω is an unbounded exterior domain.

There are several integral formulations based on integral operators K of the form

$$(Kf)(\mathbf{x}) := \int_\Gamma \kappa(\mathbf{x},\mathbf{y}) f(\mathbf{y}) d\Gamma_y, \tag{1.4}$$

where $\kappa(\mathbf{x},\mathbf{y})$ is $s(\mathbf{x},\mathbf{y})$ or some derivative with respect to $\mathbf{x}$ or $\mathbf{y}$. We give two examples.

Single-Layer Potential for a Dirichlet Problem Let the integral operator be defined by $\kappa = s$ with s from (1.2), i.e., $(Kf)(\mathbf{x}) = \frac{1}{4\pi}\int_\Gamma \frac{f(\mathbf{y})}{|\mathbf{x}-\mathbf{y}|} d\Gamma_y$ in the 3D case. Then $\Phi(\mathbf{x}) := (Kf)(\mathbf{x})$ is defined for all $\mathbf{x} \in \mathbb{R}^d$ and satisfies $\Delta\Phi = 0$ in $\mathbb{R}^d \backslash \Gamma$. In order to enforce the Dirichlet value

$$\Phi = g \qquad \text{on } \Gamma, \tag{1.5}$$

the function f has to satisfy the integral equation

$$Kf = g \quad \text{for all } \mathbf{x} \in \Gamma, \qquad \text{i.e., } \int_\Gamma \frac{f(\mathbf{y})}{|\mathbf{x}-\mathbf{y}|} d\Gamma_y = 4\pi g(\mathbf{x}) \ \text{ for all } \mathbf{x} \in \Gamma. \tag{1.6}$$

Therefore, one has to solve (a discrete version of) $Kf = g$. For the resulting solution f, the potential $\Phi = Kf$ fulfils (1.1) as well as (1.5) and can be evaluated at any point of interest.

Direct Method In (1.6) one has to solve for the unknown function f, which (indirectly) yields the solution of the Laplace problem after evaluation of $\Phi = Kf$. A direct approach is

$$\frac{1}{2}u(\mathbf{x}) = g(\mathbf{x}) + \int_\Gamma \kappa(\mathbf{x},\mathbf{y}) u(\mathbf{y}) d\Gamma_y \quad \text{with } \kappa := \frac{\partial s}{\partial n_y},\ g(\mathbf{x}) := \int_\Gamma s(\mathbf{x},\mathbf{y})\phi(\mathbf{y}) d\Gamma_y, \tag{1.7}$$

which yields the Dirichlet boundary values $u(\mathbf{x})$, $\mathbf{x} \in \Gamma$, of the interior domain with Neumann data ϕ. $s(\mathbf{x},\mathbf{y})$ is the fundamental solution from (1.2). $\kappa(\mathbf{x},\mathbf{y})$ is called double-layer kernel. The left equation in (1.7) holds for almost all $\mathbf{x} \in \Gamma$, but must be corrected by a factor corresponding to the spherical angle of an edge or corner of the surface (cf. [13]). This is important for the discretisation by collocation, but does not matter in the case of the Galerkin discretisation.

1.2.3 Discretisation by BEM

In the following, we assume the more interesting case of $d = 3$, i.e., Γ is a two-dimensional surface.

Triangulation of the Surface To begin with, assume that the surface can be represented by a union of *planar* triangles: $\Gamma = \bigcup_{t\in\mathcal{P}} t$, where the *triangulation* $\mathcal{P}$ is the set of these (closed) triangles. Usually the triangulation is required to be conforming in the sense that the intersection of two different triangles is allowed to be either empty, a node or an edge. Each $t \in \mathcal{P}$ can be produced by an affine map η_t from the unit triangle t_{unit} (vertices at $(0,0)$, $(0,1)$, $(1,0)$) onto t, i.e., $\eta_t(t_{\text{unit}}) = t$. In the following we shall assume this simple case (of course, quadrilaterals instead of triangles are possible as well).

Alternatively, the true surface can be approximated by curved triangles, i.e., Γ is replaced by $\bigcup_{t\in\mathcal{P}} \eta_t(t_{\text{unit}})$, where η_t is a more involved map producing a curved triangle.

In the BEM context the triangles are often called *panels*.

Since the panels are assumed to be closed, two different panels may overlap by their boundaries. We say that two subsets $s', s'' \subset \Gamma$ are *weakly disjoint*, if $\text{area}(s' \cap s'') = 0$. This covers the case of (completely) disjoint sets as well as the case when the boundaries overlap (but not the interior parts).

Boundary Element Space The simplest boundary element is the piecewise constant one, i.e., the boundary element space V_h consists of functions being piecewise constant on each triangle $t \in \mathcal{P}$. In the case of (continuous and) piecewise linear elements, the functions from V_h are (continuous and) piecewise affine on each triangle $t \in \mathcal{P}$. In the case of curved triangles, the piecewise affine functions on t_{unit} are mapped by η_t onto $\eta_t(t_{\text{unit}})$. Furthermore, one can consider spaces V_h of continuous or discontinuous functions which coincide with higher order polynomials on $t \in \mathcal{P}$.

Galerkin Discretisation The Galerkin discretisation of $\lambda u + Ku = \phi$ with respect to the boundary element space V_h and K from (1.4) reads

$$\begin{aligned} &\text{Find } u_h \in V_h \text{ such that} \\ &\lambda \int_\Gamma u_h(\mathbf{x}) v(\mathbf{x}) d\Gamma_x + \int_\Gamma \int_\Gamma \kappa(\mathbf{x},\mathbf{y}) u_h(\mathbf{y}) v(\mathbf{x}) d\Gamma_x d\Gamma_y = \int_\Gamma \phi(\mathbf{x}) v(\mathbf{x}) d\Gamma_x \text{ for all } v \in V_h. \end{aligned} \tag{1.8}$$

Collocation Discretisation Since (1.8) involves a double integration, often the collocation is preferred although the numerical statements about collocation are weaker. For this purpose, one defines a set $\Xi = \{\xi^i : i = 1, \ldots, n\}$ of collocation points ξ^i, where $n = \dim V_h$. For instance, in the case of piecewise constant elements, $\xi \in \Xi$ should be chosen as centroid of each $t \in \mathcal{P}$. Then the collocation discretisation of $\lambda u + Ku = \phi$ reads

$$\text{Find } u_h \in V_h \text{ such that } \lambda u_h(\xi) + \int_\Gamma \kappa(\xi,\mathbf{y}) u_h(\mathbf{y}) d\Gamma_y = \phi(\xi) \text{ for all } \xi \in \Xi. \tag{1.9}$$

Matrix Formulation Let $\mathcal{B} = \{b_1, \ldots, b_n\}$ be a basis of V_h. For instance, for piecewise constant elements, b_i is 1 on the ith triangle and 0 on each other $t \in \mathcal{P}$. In this case, we may use $t \in \mathcal{P}$ as index instead of $i = 1, \ldots, n$, i.e., the basis is $\mathcal{B} = \{b_t : t \in \mathcal{P}\}$.

In the case of discontinuous and piecewise linear elements, we have three basis functions per triangle: $\mathcal{B} = \{b_{t,k} : t \in \mathcal{P},\ k = 1, 2, 3\}$, while for continuous and piecewise linear elements, each basis function b_i is associated with a vertex $\mathbf{x}_i$ of the triangulation.

Each $u_h \in V_h$ is represented by

$$u_h = \sum_{j=1}^{n} u_j b_j, \tag{1.10}$$

where $\mathbf{u} = (u_i)_{i=1,\ldots,n}$ abbreviates the coefficient vector.

Then the solution of the *collocation* problem (1.9) is characterised by

$$\lambda \mathbf{A}\mathbf{u} + \mathbf{B}\mathbf{u} = \mathbf{f}, \tag{1.11}$$

where the matrices $\mathbf{A}, \mathbf{B}$ and the vector $\mathbf{f}$ are given by

$$\mathbf{A} = \big(b_j(\xi^i)\big)_{i=1,\ldots,n}^{j=1,\ldots,n}, \quad \mathbf{B} = \left(\int_\Gamma \kappa(\xi^i,\mathbf{y}) b_j(\mathbf{y}) d\Gamma_y\right)_{i=1,\ldots,n}^{j=1,\ldots,n}, \quad \mathbf{f} = \big(\phi(\xi^i)\big)_{i=1,\ldots,n}. \tag{1.12}$$

The *Galerkin* solution is given by (1.10) and (1.11) with

$$\begin{aligned} \mathbf{A} &= \left(\int_\Gamma b_j(\mathbf{x}) b_i(\mathbf{x}) d\Gamma_x\right)_{i=1,\ldots,n}^{j=1,\ldots,n}, \\ \mathbf{B} &= \left(\int_\Gamma \int_\Gamma \kappa(\mathbf{x},\mathbf{y}) b_j(\mathbf{y}) b_i(\mathbf{x}) d\Gamma_x d\Gamma_y\right)_{i=1,\ldots,n}^{j=1,\ldots,n}, \quad \mathbf{f} = \left(\int_\Gamma \phi(\mathbf{x}) b_i(\mathbf{x}) d\Gamma_x\right)_{i=1,\ldots,n}. \end{aligned} \tag{1.13}$$

In the case of (1.12), $\mathbf{A} = \mathbf{I}$ holds, provided that b_j is the Lagrange function. In any case, $\mathbf{A}$ is a sparse matrix which causes no problems. Differently, $\mathbf{B}$ is usually a *fully populated* matrix. Standard representation needs a storage of n^2 for all entries. The panel clustering method will reduce this size to $\mathcal{O}(n \log^q n)$, i.e., the storage will be almost linear in the dimension n. The same improvement holds for the cost of the matrix-vector multiplication.

For further details about integral equations and boundary elements see [13].

2 The Panel Clustering Method (First Version)

The panel clustering method was introduced in the mid-eighties (cf. [17]). The multipole method, which started at the same time (cf. [11]), is similar with the difference that it is more designed for point charges and requires an operator-dependent construction for the expansion functions. Quite another, but theoretically related approach is the matrix compression, which can be applied in the case of a proper wavelet discretisation (cf. [6])

The first version, which we present now, corresponds to the collocation equation (1.11), more precisely to the performance of the matrix-vector multiplication by $\mathbf{B}$, i.e., $\mathbf{u} \mapsto \mathbf{Bu}$. We recall that the ith component of $\mathbf{Bu}$ reads

$$(\mathbf{Bu})_i = \sum_{j=1}^{n} u_j \int_\Gamma \kappa(\xi^i, \mathbf{y}) b_j(\mathbf{y}) d\Gamma_y \tag{2.1}$$

(see (1.12)). For the fast evaluation of this term, we have to introduce the far field expansion (§2.1) and the panel cluster tree (§2.2).

2.1 Far Field Expansion

Consider one of the collocation points $\mathbf{x} = \xi^i$ and approximate the kernel $\kappa(\mathbf{x}, \mathbf{y})$ in a subset $\tau \subset \Gamma \backslash \{\mathbf{x}\}$ by a finite expansion of the form

$$\tilde{\kappa}(\mathbf{x}, \mathbf{y}) = \sum_{\iota \in I_m} \kappa_\iota^\tau(\mathbf{x}) \Phi_\iota(\mathbf{y}), \tag{2.2}$$

where $\Phi_\iota(\mathbf{y})$ for $\iota \in I_m$ are functions independent of $\mathbf{x}$. I_m is an index set whose size is indicated by m (see (2.3) below). The upper index τ denotes a subset of Γ in which $\tilde{\kappa}(\mathbf{x}, \cdot)$ should be a good approximation of $\kappa(\mathbf{x}, \cdot)$.

In the simplest case, $\tilde{\kappa}(\mathbf{x}, \cdot)$ is the Taylor expansion around the centre of τ up to degree $m-1$. In this case the index set I_m equals the set $\{\iota \in \mathbb{N}_0^d : \iota_1 + \ldots + \iota_d \leq m-1\}$ of multi-indices, where d is the spatial dimension. In order to make the functions $\Phi_\iota(\mathbf{y})$ independent of τ, we may choose the monomials $\Phi_\iota(\mathbf{y}) = \mathbf{y}^\iota = y_1^{\iota_1} \times \ldots \times y_d^{\iota_d}$ (expansion around zero). In the case discussed above, the number of indices of I_m is bounded by (2.3) with $C_I = 1$:

$$\# I_m \leq C_I \, m^d. \tag{2.3}$$

Further details about the approximation of κ by (2.2) will follow in §2.7.2.

2.2 Cluster Tree

Assembling several neighbouring triangles (panels) $t \in \mathcal{P}$, we can form *clusters*. The union of neighbouring clusters yield even larger clusters. This process can be continued until the complete surface Γ is obtained as largest cluster. As in Figure 2.1, the clusters may have an irregular geometric shape (they may even be unconnected). For later purpose it is favourable if clusters are rather compact, i.e., $\operatorname{area}(c)/\operatorname{diam}(c)^2$ should be large.

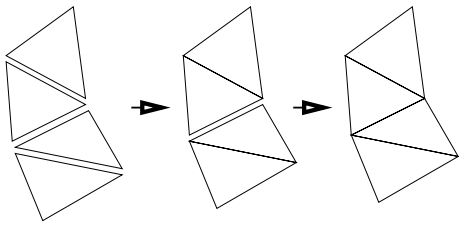

Figure 2.1: Clustering of four triangles

One may consider this process also from the opposite point of view (domain decomposition). The surface Γ is divided into smaller parts (clusters), which are divided further until only the panels remain as the trivial clusters. For a construction based on this approach see §2.7.1.

The process of clustering (or repeated domain decomposition) is represented by the *cluster tree* T. In the following definition, $\mathcal{P}$ is the set of panels and we denote the set of unions of panels by $\mathcal{S} = \{\bigcup_{t\in P'} t : P' \subset \mathcal{P}\}$. All panels $t \in \mathcal{P}$ belong to $\mathcal{S}$, but also $\Gamma = \bigcup_{t\in\mathcal{P}} t \in \mathcal{S}$.

Definition 2.1 *(a) All vertices of T belong to $\mathcal{S}$. (b) $\Gamma \in T$ is the* root *of T. (c) The* leaves *of T are the panels from $\mathcal{P}$. (d) If $\tau \in T$ is no leaf, there is a set $S(\tau)$ with at least two sons, which are weakly disjoint (cf. §1.2.3). Furthermore, $S(\tau)$ satisfies*

$$\tau = \bigcup_{\tau'\in T} \tau'. \tag{2.4}$$

Usually, $S(\tau)$ consists of exactly two sons, so that T becomes a binary tree.

Remark 2.2 *The the number of clusters $\tau \in T$ is at most $\#T \le 2\#P - 1 = 2n - 1$. The upper bound $\#T = 2n - 1$ holds for a binary tree.*

2.3 Admissible Clusters and Admissible Coverings

For the integration of $\kappa(\mathbf{x},\mathbf{y})u(\mathbf{y})$ over a cluster τ (cf. (2.1)) we shall use the expansion $\tilde\kappa(\mathbf{x},\mathbf{y})$ from (2.2) instead of κ. This requires the following condition on $\mathbf{x}$ and τ.

We call $\tau \in T$ to be an *admissible cluster* with respect to some control point $\mathbf{x} \in \mathbb{R}^d$ if

$$\operatorname{diam}(\tau) \le \eta \operatorname{dist}(\mathbf{x},\tau). \tag{2.5}$$

The parameter $\eta > 0$ will be chosen later (η will turn out to be constant independent of the panel size h). From inequality (2.5) we see: The larger the distance between $\mathbf{x}$ and the cluster, the larger the cluster may be.

A set of clusters $\mathcal{C} = \{\tau_1,\dots,\tau_s\} \subset T$ is called a *covering* (of Γ) if the clusters are weakly disjoint and satisfy

$$\Gamma = \bigcup_{j=1}^{s} \tau_j. \tag{2.6}$$

There are two trivial coverings: $\mathcal{C} = \{\Gamma\}$ is the coarsest one. The finest is $\mathcal{C} = \mathcal{P}$. In the first case the cluster is as large as possible (but the number of cluster is minimum), in the second case the clusters are as small as possible (but their number is maximum).

In the following, we are looking for a covering which (from the computational point of view) should consist of a small number of clusters and which, on the other hand, should be admissible. This leads us to the following definition.

Definition 2.3 *We call $\mathcal{C} = \{\tau_1,\dots,\tau_\sigma\} \subset T$ an* admissible covering *(of Γ) with respect to $\mathbf{x}$ if it is a covering satisfying (2.6) and*

$$\text{either } \tau_j \in \mathcal{P} \text{ or } \tau_j \text{ is admissible with respect to } \mathbf{x}. \tag{2.7}$$

Remark 2.4 *(a) If $\mathbf{x} \in \Gamma$, there is no covering (2.6) of Γ consisting of admissible clusters only. (b) Condition (2.7) states that non-admissible clusters are panels.*

The number of clusters in $\mathcal{C}$ should be as small as possible. The optimum is discussed in

Proposition 2.5 *For each $\mathbf{x} \in \mathbb{R}^d$ there is a unique admissible covering $\mathcal{C}(\mathbf{x})$ with respect to $\mathbf{x}$ with minimum number $n_{\mathcal{C}}(\mathbf{x}) := \#\mathcal{C}(\mathbf{x})$ of clusters. $\mathcal{C}(\mathbf{x})$ is called the* minimum admissible covering *with respect to $\mathbf{x}$.*

The minimum admissible covering $\mathcal{C}(\mathbf{x})$ with respect to $\mathbf{x}$ can easily be computed by

$$\mathcal{C} := \emptyset;\ \mathit{Divide}(\Gamma,\mathcal{C});\ \textbf{comment}\ \text{the result is } \mathcal{C} = \mathcal{C}(\mathbf{x}); \tag{2.8a}$$

where *Divide* is the recursive procedure

```
procedure Divide(τ,C); comment τ ∈ T is a cluster, C is a subset of T;
begin  if τ is admissible with respect to x then C := C ∪ {τ}
       else  if τ ∈ P then C := C ∪ {τ}
             else for all τ' ∈ S(τ) do Divide(τ',C)
end;
```
(2.8b)

2.4 Algorithm for the Matrix-Vector Multiplication

2.4.1 Partition into Near and Far Field

Instead of computing the matrix entries, we compute the *far field coefficients* in Phase I. In Phase II we evaluate an approximation of Ku_h at $\mathbf{x} \in \mathbb{R}^d$ (e.g., at $\mathbf{x} = \xi^i \in \Xi$). Repeating (2.1), we recall that the desired result of the matrix-vector multiplication $\mathbf{v} = \mathbf{Bu}$ is

$$v_i = \sum_{j=1}^{n} u_j \int_\Gamma \kappa(\xi^i, \mathbf{y}) b_j(\mathbf{y}) d\Gamma_y. \tag{2.9}$$

Let $\mathcal{C}(\xi^i)$ be the minimum admissible covering determined in (2.8b). We split $\mathcal{C}(\xi^i)$ into a *near field* and a *far field* part defined by

$$\begin{array}{rcl} \mathcal{C}_{\text{near}}(\xi^i) & := & \{\tau \in \mathcal{C}(\xi^i) : \tau \text{ is not admissible}\}, \\ \mathcal{C}_{\text{far}}(\xi^i) & := & \{\tau \in \mathcal{C}(\xi^i) : \tau \text{ is admissible}\}. \end{array}$$

All $\tau \in \mathcal{C}_{\text{near}}(\xi^i)$ are panels (cf. Remark 2.4b). The integral in (2.9) can be written as

$$\int_\Gamma \ldots = \sum_{\tau \in \mathcal{C}_{\text{near}}(\xi^i)} \int_\tau \ldots + \sum_{\tau \in \mathcal{C}_{\text{far}}(\xi^i)} \int_\tau \ldots .$$

This induces an analogous splitting of v_i from (2.9) into

$$v_i = v_i^{\text{near}} + v_i^{\text{far}}.$$

The part v_i^{far} will be approximated by $\tilde{v}_i^{\text{far}}$ in §2.4.3. Note that the splitting depends on i. Another i' yields another collocation point $\xi^{i'}$ and another splitting into $\mathcal{C}_{\text{near}}(\xi^{i'})$ and $\mathcal{C}_{\text{far}}(\xi^{i'})$.

2.4.2 Near Field Part

The near field part of v_i is computed exactly (or with sufficiently accurate quadrature):

$$v_i^{\text{near}} = \sum_{t \in \mathcal{C}_{\text{near}}(\xi^i)} \sum_j u_j \int_t \kappa(\xi^i, \mathbf{y}) b_j(\mathbf{y}) d\Gamma_y. \tag{2.10}$$

Since the support of the basis functions b_j is small, there is only a constant number of indices j such that a panel $t \in \mathcal{C}_{\text{near}}(\xi^i)$ intersects with $\text{supp}(b_j)$. Hence the sum $\sum_j$ has only $\mathcal{O}(1)$ terms. The number of panels in $\mathcal{C}_{\text{near}}(\xi^i)$ turns out to be bounded by $\mathcal{O}(\log n)$.

2.4.3 Far Field Part

Replacing the exact kernel κ in the definition of v_i^{far} by $\tilde{\kappa}(\mathbf{x}, \mathbf{y})$ from (2.2), we obtain

$$\begin{array}{rcl} \tilde{v}_i^{\text{far}} & := & \sum_{\tau \in \mathcal{C}_{\text{far}}(\xi^i)} \sum_j u_j \int_\tau \tilde{\kappa}(\xi^i, \mathbf{y}) b_j(\mathbf{y}) d\Gamma_y \\ & = & \sum_{\tau \in \mathcal{C}_{\text{far}}(\xi^i)} \sum_j u_j \int_\tau \sum_{\iota \in I_m} \kappa_\iota^\tau(\xi^i) \Phi_\iota(\mathbf{y}) b_j(\mathbf{y}) d\Gamma_y \, . \end{array}$$

Summation and integration can be interchanged:

$$\tilde{v}_i^{\text{far}} = \sum_{\tau \in \mathcal{C}_{\text{far}}(\xi^i)} \sum_{\iota \in I_m} \kappa_\iota^\tau(\xi^i) \sum_j u_j \int_\tau \Phi_\iota(\mathbf{y}) b_j(\mathbf{y}) d\Gamma_y. \tag{2.11}$$

The following integrals are called *far field coefficients*,

$$J_\tau^\iota(b_j) := \int_\tau \Phi_\iota(\mathbf{y}) b_j(\mathbf{y}) d\Gamma_y \qquad (\iota \in I_m,\ \tau \in T,\ 1 \le j \le n). \tag{2.12}$$

Of particular interest are the far field coefficients corresponding to panels. These are the only ones to be evaluated in the first phase:

$$J_t^\iota(b_j) := \int_t \Phi_\iota(\mathbf{y}) b_j(\mathbf{y}) d\Gamma_y \qquad (\iota \in I_m,\ t \in \mathcal{P},\ 1 \le j \le n). \tag{2.13}$$

Remark 2.6 *(a) The coefficients $J_t^\iota(b_j)$ are independent of the special vector* $\mathbf{u}$ *in the matrix-vector multiplication* $\mathbf{v} = \mathbf{Bu}$*. (b) There is only a fixed number of panels t intersecting with the support of b_j; otherwise, $J_t^\iota(b_j) = 0$. The number of non-zero coefficients $J_t^\iota(b_j)$ is $\mathcal{O}(n\,\#I_m)$.*

As soon as the far field coefficients (2.13) are computed, the quantities

$$J_t^\iota := \sum_j u_j J_t^\iota(b_j) \qquad (\iota \in I_m,\ t \in \mathcal{P}) \tag{2.14}$$

can be summed up by $\mathcal{O}(n\,\#I_m)$ additions. For $\tau \in T\backslash\mathcal{P}$, we exploit the tree structure:

$$J_\tau^\iota = \sum_{\tau' \in S(\tau)} J_{\tau'}^\iota \qquad \text{for } \tau \in T\backslash\mathcal{P}. \tag{2.15}$$

The coefficients J_τ^ι represent the sum

$$J_\tau^\iota = \sum_j u_j J_\tau^\iota(b_j) = \int_\tau \Phi_\iota(\mathbf{y}) \sum_j u_j b_j(\mathbf{y})\, d\Gamma_y\,.$$

Hence the quantities $\tilde{v}_i^{\mathrm{far}}$ can be computed from the simple sum

$$\tilde{v}_i^{\mathrm{far}} = \sum_{\tau \in \mathcal{C}_{\mathrm{far}}(\xi^i)} \sum_{\iota \in I_m} \kappa_\iota^\tau(\xi^i)\ J_\tau^\iota \tag{2.16}$$

(see (2.11)). Since the number of clusters $\tau \in \mathcal{C}_{\mathrm{far}}(\xi^i)$ is expected to be much smaller than the number n of all panels, the representation (2.16) should be advantageous.

2.5 The Additional Quadrature Error

The replacement of v_i^{far} by $\tilde{v}_i^{\mathrm{far}}$ can be regarded as an additional quadrature error. The error of the expansion (2.2) depends on the order m and the cluster containing $\mathbf{y}$. The exact requirements on $\tilde{\kappa}$ are as follows.

Assumption 2.7 *Let $\eta_0 \in (0,1)$ and a ball $B \subset \mathbb{R}^d$ be given. There are constants C_1, C_2 such that for all $0 < \eta < \eta_0 < 1$ and $m \in \mathbb{N}$ there are expansions $\tilde{\kappa}$ of the form (2.2) satisfying*

$$|\kappa(\mathbf{x},\mathbf{y}) - \tilde{\kappa}(\mathbf{x},\mathbf{y})| \le C_1(C_2\eta)^m |\kappa(\mathbf{x},\mathbf{y})| \quad \textit{for all } \mathbf{y} \in B \textit{ and } \operatorname{diam} B \le \eta \operatorname{dist}(\mathbf{x}, B)\,. \tag{2.17}$$

Inequality (2.17) provides an estimation of the relative error of $\tilde{\kappa}$. The proof of (2.17) in [18] uses a Taylor expansion $\tilde{\kappa}$ with respect to $\mathbf{y}$ for standard examples and determines the values of C_1,C_2,η_0. The estimation in (2.17) by $\kappa(\mathbf{x},\mathbf{y})$ on the right-hand side makes for instance sense for positive kernels like $\kappa(\mathbf{x},\mathbf{y}) = \frac{1}{4\pi|\mathbf{x}-\mathbf{y}|}$. In §2.7.3 it will become obvious that (2.17) can also be obtained for the double-layer kernel (see (1.7)) although its sign changes.

By (2.17) the error is of order $\mathcal{O}(\eta^m)$. In order to make the error equal to the consistency error $\mathcal{O}(h^\varkappa)$ (h: panel size), we choose η in (2.5) by

$$\eta = \eta(n) := \mathcal{O}((n^{-\varkappa/(d-1)})^{1/m}) < \eta_0, \quad \varkappa < m. \tag{2.18}$$

Then $\mathcal{O}(h^m)$ equals $\mathcal{O}(h^\varkappa)$. Since in (2.20) m will be chosen as $\mathcal{O}(\log n)$, the quantity η becomes independent of h.

2.6 Complexity of the Algorithm

2.6.1 Choice of Parameters

The complexity of the algorithm depends mainly on the number $n_{\mathcal{C}}(\mathbf{x})$ of clusters in the minimum admissible covering $\mathcal{C}(\mathbf{x})$. Under natural conditions described in [18], where also details of the proofs can be found, there is a constant $C_{\mathcal{C}}$ with

$$n_{\mathcal{C}}(\mathbf{x}) \le n_{\mathcal{C}}(\eta, n) := C_{\mathcal{C}}(\tfrac{1}{\eta})^{d-1} \log(2 + \eta^{d-1}\#\mathcal{P}) \qquad \text{for all } \mathbf{x} \in \mathbb{R}^d \tag{2.19}$$

with η from (2.5). The logarithmic factor can even be omitted if $\mathbf{x} \notin \Gamma$.

Inserting (2.18) into (2.19) and using $\#\mathcal{P} = \mathcal{O}(n)$, we obtain the following estimate for the number $n_{\mathcal{C}}(\mathbf{x})$ of clusters in $\mathcal{C}(\mathbf{x})$:

$$n_{\mathcal{C}}(\mathbf{x}) \le n_{\mathcal{C}}(\eta, n) = C'_{\mathcal{C}} n^{\varkappa/m} \log(2 + C_d n^{1-\varkappa/m}) \qquad \text{for all } \mathbf{x} \in \mathbb{R}^d.$$

The optimal choice of the expansion order m turns out to be

$$m := \left\lfloor \tfrac{\varkappa}{d+1} \log n \right\rfloor \qquad (\lfloor x \rfloor := \text{largest integer } i \text{ with } i \le x). \tag{2.20}$$

Then $n^{\varkappa/m}$ (and η) is a constant and we obtain the estimate

$$n_{\mathcal{C}}(\mathbf{x}) \le C \log n. \tag{2.21}$$

Therefore, we have to deal with only $\mathcal{O}(\log n)$ clusters instead of n panels.

2.6.2 Operation Count

While Phase I has to be performed only once for initialisation, Phase II has to be repeated for every matrix-vector multiplication.

Phase I (a) Algorithm (2.8a,b) (computing the minimum admissible covering $\mathcal{C}(\mathbf{x})$) requires $\mathcal{O}(n_{\mathcal{C}}(\mathbf{x}))$ operations per point $\mathbf{x}$. Because of (2.21) and since there are n different collocation points $\mathbf{x} = \xi^i$, the total amount of work in this part is $\mathcal{O}(n \log n)$.

(b) The computation of v_i^{near} in (2.10) needs $\mathcal{O}(\log n)$ evaluations of integrals of the form $\int_t \kappa(\xi^i, \mathbf{y}) b_j(\mathbf{y}) d\Gamma_y$ per index i, therefore in total $\mathcal{O}(n \log n)$ evaluations.

(c) The far field coefficients $J_t^\iota(b_j)$ ($t \in \mathcal{P}$, $\iota \in I_m$) can be computed by $\mathcal{O}(n \log^d n)$ operations (cf. (2.3), (2.20)) and require $\mathcal{O}(n \log^d n)$ evaluations (or approximations) of integrals of the form $\int_t \Phi_\iota(\mathbf{y}) b_j(\mathbf{y}) d\Gamma_y$.

(d) The number of coefficients $\kappa_\iota^\tau(\xi^i)$ to be evaluated for $\tau \in \mathcal{C}_{\text{far}}(\xi^i)$, $\iota \in I_m$, $1 \le i \le n$ equals $\mathcal{O}(n \log^{d+1} n)$.

Phase II (a) The far field coefficients J_τ^ι for the nontrivial clusters $\tau \in T \backslash \mathcal{P}$ and all indices $\iota \in I_m$ can be summed up in $\mathcal{O}(n \# I_m) = \mathcal{O}(n \log^d n)$ additions (see (2.15)).

(b) The final summation in (2.16) requires only $\mathcal{O}(n \log^{d+1} n)$ additions.

Theorem 2.8 *(a) The data computed in Phase I and the quantities from Phase II require a storage of size $\mathcal{O}(n \log^{d+1} n)$ data. (b) Each matrix-vector multiplication $\mathbf{u} \mapsto \mathbf{Bu}$ can be approximated up to an error of size $\mathcal{O}(h^\varkappa)$ by $\mathcal{O}(n \log^{d+1} n)$ operations.*

Concerning the storage in Phase I, we remark that only the coverings $\mathcal{C}(\xi^i)$, the non-zero integrals $\int_t \kappa(\xi^i, \mathbf{y}) b_j(\mathbf{y}) d\Gamma_y$ from (2.10), the expansion coefficients $\kappa_\iota^\tau(\xi^i)$, and the far field coefficients $J_t^\iota(b_j)$ are to be stored.

The costs for Phase I can be further reduced if several panels are geometrically similar.

2.7 Some Implementational Details

2.7.1 Construction of the Cluster Tree

In the following, we describe the construction of the cluster tree T by means of bounding boxes. This method is in particular suited for elements in a domain $\Omega \subset \mathbb{R}^d$. The application to surfaces will be discussed at the end of this section.

Associate every element t with its centroid denoted by $\mathbf{z}_t$ with the coordinates $z_{t,k}$ ($k = 1, \ldots, d$). Let Z be the set of these points. The smallest bounding box $Q \subset \mathbb{R}^d$ containing all $\mathbf{z}_t$ is given by

$$Q = [a_1, b_1] \times \ldots \times [a_d, b_d], \text{ where } a_k := \min\{z_k : \mathbf{z} \in Z\},\ b_k := \max\{z_k : \mathbf{z} \in Z\}. \tag{2.22}$$

Choose $k \in \{1, \ldots, d\}$ such that the side length $b_k - a_k$ is maximum and divide Q into

$$Q^I = [a_1, b_1] \times \ldots \times \left[a_k, a_k + \tfrac{b_k - a_k}{2}\right) \times \ldots \times [a_d, b_d],$$
$$Q^{II} = [a_1, b_1] \times \ldots \times \left[a_k + \tfrac{b_k - a_k}{2}, b_k\right] \times \ldots \times [a_d, b_d].$$

This gives rise to a partition of Z into $Z' := Z \cap Q^I$ and $Z'' := Z \cap Q^{II}$. The procedure can be repeated recursively: Determine the bounding box Q' of Z' (it may be smaller than Q^I!) and split Q' into $Q'^{,I}$ and $Q'^{,II}$ and accordingly Z'. The recursion stops when the resulting subset of Z contains only one point. Obviously, the construction produces a binary tree T_Z starting with the root Z. Any vertex of T_Z is a subset Z^* of Z. Since each $\mathbf{z} \in Z^*$ corresponds to exactly one panel $t = t_{\mathbf{z}} \in \mathcal{P}$, the union $\bigcup_{\mathbf{z}\in Z^*} t_{\mathbf{z}}$ describes a cluster. In this way, the tree T_Z can be transferred into the desired cluster tree.

Figure 2.2 shows the bisection process in the two-dimensional case.

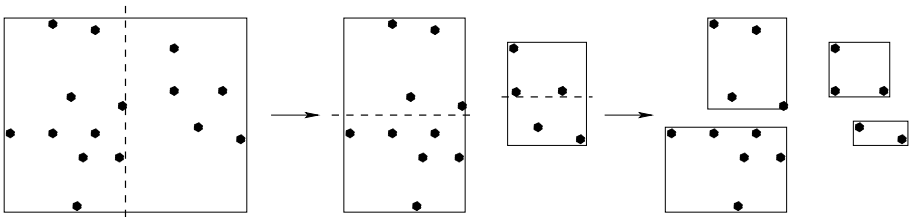

Figure 2.2: The bounding box to the left containing the points $\mathbf{z}_t$ is divided into two parts in z_1-direction. In the second step, the new bounding boxes are divided in z_2-direction

For BEM, a modification is of interest. As soon as the corresponding cluster is close to some (tangent) hyperplane, the coordinates of the bounding box can be rotated so that $d-1$ coordinates are in the hyperplane, while the dth coordinate is the normal direction.

2.7.2 Far Field Expansion by Polynomial Interpolation

In (2.2), $\tilde{\kappa}(\mathbf{x},\mathbf{y}) = \sum_{\iota\in I_m} \kappa_\iota^\tau(\mathbf{x})\Phi_\iota(\mathbf{y})$ describes the approximation of $\kappa(\mathbf{x},\mathbf{y})$ in the cluster τ for a *fixed* collocation point $\mathbf{x}$. Let $d_m = \#I_m$ denote the dimension of polynomials of total degree $m-1$. Choose d_m interpolation points ζ_i^τ and let $\tilde{\kappa}(\mathbf{x},\mathbf{y})$ be the polynomial interpolating $\kappa(\mathbf{x},\mathbf{y})$. It has the representation $\sum_{i=1}^{d_m} \kappa(\mathbf{x},\zeta_i^\tau)L_i^\tau(\mathbf{y})$, where $L_i^\tau(\mathbf{y})$ denotes the Lagrange polynomials with the property $L_i^\tau(\zeta_j^\tau) = \delta_{ij}$ (Kronecker symbol). Expanding L_i into monomials, one obtains the desired representation (2.2).

Concerning the choice of interpolation points, one is not restricted to the cluster $\tau \subset \Gamma \subset \mathbb{R}^d$. Instead one can first define suitable quadrature points ζ_i in the unit cube $C = [-1/2, 1/2]^d$. Given a cluster τ with centre $\mathbf{z}_\tau$, consider the cube $C_\tau := \mathbf{z}_\tau + \mathrm{diam}(\tau)C$ and use the corresponding quadrature points $\zeta_i^\tau = \mathbf{z}_\tau + \mathrm{diam}(\tau)\zeta_i$. The interpolation polynomial converges to the Taylor polynomial if all interpolation points ζ_i^τ tend to the centre $\mathbf{z}_\tau$.

The latter approach requires that $\kappa(\mathbf{x},\cdot)$ is defined on C_τ. This holds, e.g., for kernels from §1.2.1, but not for kernels involving normal derivatives as the double layer kernel, since the normal derivative is defined on Γ only. The remedy is given in the next subsection.

2.7.3 Far Field Expansion for Kernels with Normal Derivatives

The double layer kernel for the Laplace problem is $\frac{\partial}{\partial n_y}\frac{1}{4\pi|\mathbf{x}-\mathbf{y}|} = \frac{1}{4\pi}\frac{\langle \mathbf{x}-\mathbf{y}, \mathbf{n}(\mathbf{y})\rangle}{|\mathbf{x}-\mathbf{y}|^2}$ (cf. (1.7)). One possibility is to approximate $1/|\mathbf{x}-\mathbf{y}|^2$ by an expression of the form $\sum_\iota \kappa_\iota^*(\mathbf{x})\Phi_\iota(\mathbf{y})$. Then $\frac{1}{4\pi}\frac{\langle \mathbf{x}-\mathbf{y}, \mathbf{n}(\mathbf{y})\rangle}{|\mathbf{x}-\mathbf{y}|^2}$ is approximated by $\sum_{j=1}^d \sum_\iota \{x_j\kappa_\iota^*(\mathbf{x})\}\{n_j(\mathbf{y})\Phi_\iota(\mathbf{y})\} - \sum_\iota \kappa_\iota^*(\mathbf{x})\{\langle \mathbf{y}, \mathbf{n}(\mathbf{y})\rangle \Phi_\iota(\mathbf{y})\}$ and the latter expression is again of the form (2.2). Note that nonsmooth surfaces yielding nonsmooth normal directions $\mathbf{n}(\mathbf{y})$ cause no difficulty. Furthermore, the relative error estimate (2.17) can be shown (the error becomes zero if $\frac{1}{4\pi}\frac{\langle \mathbf{x}-\mathbf{y}, \mathbf{n}(\mathbf{y})\rangle}{|\mathbf{x}-\mathbf{y}|^2} = 0$ due to $\mathbf{x}-\mathbf{y} \perp \mathbf{n}(\mathbf{y})$).

The disadvantage of the described approach is the fact that the number of terms is multiplied by the factor 4. This can be avoided by approximating $\frac{1}{4\pi|\mathbf{x}-\mathbf{y}|}$ by $\sum_\iota \kappa_\iota(\mathbf{x})\Phi_\iota^*(\mathbf{y})$ and forming its normal derivative: $\sum_\iota \kappa_\iota(\mathbf{x})\frac{\partial}{\partial n_y}\Phi_\iota^*(\mathbf{y})$, which gives (2.2) with $\Phi_\iota(\mathbf{y}) := \frac{\partial}{\partial n_y}\Phi_\iota^*(\mathbf{y})$.

The latter approach is in particular helpful when the Lamé equation is treated. Note that $(\frac{\partial}{\partial x_i}\frac{\partial}{\partial x_j}|\mathbf{x}-\mathbf{y}|)_{i,j=1,\ldots,3} = \frac{1}{|\mathbf{x}-\mathbf{y}|}\mathbf{I} - \frac{(\mathbf{x}-\mathbf{y})(\mathbf{x}-\mathbf{y})^\top}{|\mathbf{x}-\mathbf{y}|^3}$; hence, the expansion of the scalar function $|\mathbf{x}-\mathbf{y}|$ has to be differentiated.

2.8 Modification: Approximations with Basis Transforms

In $\tilde{\kappa}(\mathbf{x},\mathbf{y}) = \sum_{\iota\in I_m} \kappa_\iota^\tau(\mathbf{x})\Phi_\iota(\mathbf{y})$ (cf. (2.2)) we required $\Phi_\iota(\mathbf{y})$ to be independent of τ. This fact was used in (2.15): The quadrature results of $J_{\tau'}^\iota(b_j) = \int_{\tau'} \Phi_\iota(\mathbf{y})b_j(\mathbf{y})d\Gamma_y$ for the sons $\tau' \in S(\tau)$ could be used to get $J_\tau^\iota(b_j) = \int_\tau \Phi_\iota(\mathbf{y})b_j(\mathbf{y})d\Gamma_y$ as their sum.

However, a global basis $\{\Phi_\iota(\mathbf{y}) : \iota \in I_m\}$ has numerical disadvantages. Considering, e.g., polynomials, one likes to have locally defined bases $\{\Phi_\tau^\iota(\mathbf{y}) : \iota \in I_m\}$ for each cluster $\tau \in T$. Since these different bases span the same spaces, there are transformations of the form

$$\Phi_\tau^\iota(\mathbf{y}) = \sum_{\lambda\in I_m} \omega_{\tau,\tau'}^{\iota,\lambda}\Phi_{\tau'}^\lambda(\mathbf{y}) \qquad \text{for } \tau \in T \text{ and } \tau' \in S(\tau). \tag{2.23}$$

We redefine

$$J_\tau^\iota(b_j) := \int_\tau \Phi_\tau^\iota(\mathbf{y})b_j(\mathbf{y})d\Gamma_y \tag{2.24}$$

using the τ-dependent basis $\{\Phi_\tau^\iota(\mathbf{y}) : \iota \in I_m\}$. The computation starts at the leaves (panels): $J_t^\iota(b_j)$ is computed for all $t \in \mathcal{P}$. Due to (2.23), we have $J_\tau^\iota(b_j) = \sum_{\tau'\in S(\tau)} \sum_{\lambda\in I_m} \omega_{\tau,\tau'}^{\iota,\lambda} J_{\tau'}^\lambda(b_j)$ instead of (2.12). We store only $J_t^\iota(b_j)$ for $t \in \mathcal{P}$ and compute for a given vector $\mathbf{u}$ the quantities $J_t^\iota = \sum_j u_j J_t^\iota(b_j)$ as in (2.14). However, the formula for $J_\tau^\iota = \sum_j u_j J_\tau^\iota(b_j)$ has now to use (2.24) and reads

$$J_\tau^\iota = \sum_{\tau'\in S(\tau)} \sum_{\lambda\in I_m} \omega_{\tau,\tau'}^{\iota,\lambda} J_{\tau'}^\lambda$$

instead of (2.15). These J_τ^ι can now be used in (2.16) to obtain $\tilde{v}_i^{\text{far}}$.

Concerning the coefficients $\omega_{\tau,\tau'}^{\iota,\lambda}$, we return to the basis of Lagrange functions L_i^τ introduced in §2.7.2. In that case, $\omega_{\tau,\tau'}^{\iota,\lambda} = L_\lambda^{\tau'}(\zeta_\iota^\tau)$ involves nothing than the evaluation of the τ'-basis functions at the interpolation points associated to τ.

Another obvious basis are the monomials $(\mathbf{y} - \mathbf{z}_\tau)^\iota$ centred around a midpoint $\mathbf{z}_\tau$ of the cluster τ. In this case, (2.23) describes the re-expansion of polynomials centred at $\mathbf{z}_{\tau'}$ around the new centre $\mathbf{z}_\tau$.

3 The Panel Clustering Method (Second Version)

The previous version of the panel clustering method is completely row-oriented. For each row index i, we compute the component v_i of $\mathbf{v} = \mathbf{Bu}$ by means of a covering $\mathcal{C}(\xi^i)$ which may change with i. As a consequence, the kernel $\kappa(\mathbf{x},\mathbf{y}) = \kappa(\xi^i,\mathbf{y})$ is a function of $\mathbf{y}$ only and (2.2) describes an expansion with respect to $\mathbf{y}$.

In the following, we try to determine a version in which the $\mathbf{x}$- and $\mathbf{y}$-directions are equally treated. This is in particular more appropriate for the Galerkin discretisation (1.8).

The tree T from §2.2 was introduced to describe decompositions of Γ. Now we consider the product $\Gamma\times\Gamma$ and determine a corresponding (second) tree T_2 in §3.1.1. The vertices of T_2 are products $\tau\times\sigma \subset \Gamma\times\Gamma$ of clusters $\tau,\sigma \in T$. The kernel $\kappa(\mathbf{x},\mathbf{y})$ will be approximated by a special separable expansion (3.3) for $(\mathbf{x},\mathbf{y}) \in \tau\times\sigma$.

3.1 The Tree T_2 of Products of Clusters

3.1.1 Definition

Let the cluster tree T be defined as in §2.2. The second tree T_2 is constructed from T as follows. We use the symbol b for the vertices of T_2. While $S(\tau)$ denotes the set of sons of $\tau \in T$, the sons of $b \in T_2$ form the set $S_2(b)$.

Definition 3.1 *(a) T_2 is a subset of $T\times T$, i.e., each vertex of T_2 is a product $\tau\times\sigma$ of two clusters $\tau,\sigma \in T$. (b) $\Gamma\times\Gamma \in T_2$ is the root of the tree. (c) It remains to construct the set $S_2(b)$ of sons of any $b = \tau\times\sigma \in T_2$:*

$$S_2(b) := \begin{cases} \{\tau'\times\sigma' : \sigma' \in S(\sigma), \tau' \in S(\tau)\} & \text{if neither } \sigma \text{ nor } \tau \text{ are leaves of } T_2, \\ \{\tau\times\sigma' : \sigma' \in S(\sigma)\} & \text{if } \tau \text{ is a leaf of } T_2 \text{ but not } \sigma, \\ \{\tau'\times\sigma : \tau' \in S(\tau)\} & \text{if } \sigma \text{ is a leaf of } T_2 \text{ but not } \tau, \\ \emptyset & \text{if } \tau \text{ and } \sigma \text{ are leaves of } T_2. \end{cases}$$

The last case, $S_2(b) = \emptyset$ is equivalent to saying that b is a leaf in T_2. Note that (b) defines a first vertex in T_2, while by (c) one gets recursively new vertices belonging to T_2. In this way, T_2 is completely defined by T. In particular, only the tree structure of T has to be stored.

Remark 3.2 *The tree T_2 has the same properties as T in (2.4): For any $b \in T_2$ being not a leaf, the sons $b' \in S_2(b)$ are weakly disjoint and $b = \bigcup_{b' \in S_2(b)} b'$. The leaves of T_2 are of the form $\tau \times \sigma$ with panels $\tau, \sigma \in \mathcal{P}$.*

3.1.2 Admissibility, Covering $\mathcal{C}_2$

Let $\eta > 0$ be the same parameter as in (2.5). A product $b = \tau \times \sigma \in T_2$ is called *admissible* if

$$\max\{\operatorname{diam}(\tau), \operatorname{diam}(\sigma)\} \le \eta \operatorname{dist}(\tau, \sigma). \tag{3.1}$$

As in Definition 2.3 we define a covering of $\Gamma \times \Gamma$.

Definition 3.3 *(a) A* covering $\mathcal{C}_2 \subset T_2$ *is a subset with pairwise weakly disjoint $b \in \mathcal{C}_2$ such that $\bigcup_{b \in \mathcal{C}_2} b = \Gamma \times \Gamma$. (b) An* admissible covering $\mathcal{C}_2$ *is a covering such that all $b \in \mathcal{C}_2$ are either admissible or a leaf.*

Again we are looking for a minimum admissible covering $\mathcal{C}_2$, which is obtained by (3.2a) using *Divide*2 from (3.2b),

$$\mathcal{C}_2 := \emptyset;\ \mathit{Divide}2(\Gamma \times \Gamma, \mathcal{C}_2); \tag{3.2a}$$

```
procedure Divide2(b, C2); comment b ∈ T2, C2 ⊂ T2;
begin  if b is admissible then C2 := C2 ∪ {b}
       else  if b is a leaf of T2 then C2 := C2 ∪ {b}
             else for all b' ∈ S2(b) do Divide2(b', C2)
end;
```
(3.2b)

In the following, $\mathcal{C}_2$ denotes the minimum admissible covering obtained by (3.2a-b).

3.2 Kernel Expansion

We split $\mathcal{C}_2$ into a far field $\mathcal{C}_2^{\text{far}} := \{b \in \mathcal{C}_2 : b \text{ is admissible}\}$ and near field $\mathcal{C}_2^{\text{near}} := \{b \in \mathcal{C}_2 : b \text{ is not admissible}\}$. In the latter case, b is a leaf of T_2. Due to the admissibility condition (3.1), the kernel function $\kappa(\mathbf{x}, \mathbf{y})$ allows an expansion with respect to $\mathbf{x} \in \tau$ *and* $\mathbf{y} \in \sigma$, when $b \in \mathcal{C}_2^{\text{far}}$. For this purpose, we introduce a basis $\{\Phi_\tau^\nu : \nu \in I_m\}$ for each cluster $\tau \in T$, which is applied with respect to $\mathbf{x}$ and $\mathbf{y}$.

Given $b = \tau \times \sigma \in \mathcal{C}_2^{\text{far}}$, we approximate $\kappa(\mathbf{x}, \mathbf{y})$ by an expression $\kappa_b(\mathbf{x}, \mathbf{y})$ of the form

$$\kappa_b(\mathbf{x}, \mathbf{y}) := \sum_{\nu \in I_m} \sum_{\mu \in I_m} \kappa_{\nu,\mu}^b \Phi_\tau^\nu(\mathbf{x}) \Phi_\sigma^\mu(\mathbf{y}) \qquad \text{for } (\mathbf{x}, \mathbf{y}) \in b = \tau \times \sigma \in \mathcal{C}_2^{\text{far}}. \tag{3.3}$$

An example of such an expression is the Taylor expansion with respect to $(\mathbf{x}, \mathbf{y})$ around the centres $(\mathbf{z}_\tau, \mathbf{z}_\sigma)$ of τ and σ. Then $\Phi_\tau^\nu(\mathbf{x})$ is the monomial $(\mathbf{x} - \mathbf{z}_\tau)^\nu$, where $\nu \in I_m$ belongs to the same set of multi-indices as for the Taylor expansion in §2.1.

The coefficients $\kappa_{\nu,\mu}^b$ in (3.3) form a $d_m \times d_m$ matrix $K^b = \left(\kappa_{\nu,\mu}^b\right)_{\nu,\mu \in I_m}$, where $d_m := \#I_m$.

3.3 Matrix-Vector Multiplication for the Galerkin Discretisation

We recall the Galerkin discretisation (1.8) and the matrix formulation (1.13) involving the matrix $\mathbf{B}$. The ith component of $\mathbf{v} = \mathbf{B}\mathbf{u}$ is

$$v_i = \sum_{j=1}^n u_j \int_\Gamma \int_\Gamma \kappa(\mathbf{x}, \mathbf{y}) b_j(\mathbf{y}) b_i(\mathbf{x}) d\Gamma_x d\Gamma_y.$$

The covering $\mathcal{C}_2$ allows to replace the integration over $\Gamma \times \Gamma$ by the sum of integrals over $b \in \mathcal{C}_2$,

$$v_i = \sum_{b = \tau \times \sigma \in \mathcal{C}_2} \sum_{j=1}^n u_j \int_\tau \int_\sigma \kappa(\mathbf{x}, \mathbf{y}) b_j(\mathbf{y}) b_i(\mathbf{x}) d\Gamma_x d\Gamma_y.$$

If $b = \tau \times \sigma \in \mathcal{C}_2^{\rm near}$, the expression remains unchanged and yields the near field part $v_i^{\rm near}$. For $b = \tau \times \sigma \in \mathcal{C}_2^{\rm far}$, we replace $\kappa(\mathbf{x}, \mathbf{y})$ by $\kappa_b(\mathbf{x}, \mathbf{y})$ from (3.3),

$$\begin{aligned} v_i^{\rm far} &= \sum_{b=\tau\times\sigma\in\mathcal{C}_2^{\rm far}} \sum_{j=1}^{n} u_j \int_\tau \int_\sigma \underbrace{\sum_{\nu\in I_m} \sum_{\mu\in I_m} \kappa_{\nu,\mu}^b \Phi_\tau^\nu(\mathbf{x}) \Phi_\sigma^\mu(\mathbf{y})}_{=\kappa_b(\mathbf{x},\mathbf{y})} \, b_j(\mathbf{y}) b_i(\mathbf{x}) d\Gamma_x d\Gamma_y \\ &= \sum_{b=\tau\times\sigma\in\mathcal{C}_2^{\rm far}} \sum_{j=1}^{n} u_j \sum_{\nu\in I_m} \sum_{\mu\in I_m} \kappa_{\nu,\mu}^b \left(\int_\tau \Phi_\tau^\nu(\mathbf{x}) b_i(\mathbf{x}) d\Gamma_x \right) \left(\int_\sigma \Phi_\sigma^\mu(\mathbf{y}) b_j(\mathbf{y}) d\Gamma_y \right) \\ &= \sum_{b=\tau\times\sigma\in\mathcal{C}_2^{\rm far}} \sum_{j=1}^{n} u_j \sum_{\nu\in I_m} \sum_{\mu\in I_m} \kappa_{\nu,\mu}^b \, J_\tau^\nu(b_i) \, J_\sigma^\mu(b_j) = \sum_{b=\tau\times\sigma\in\mathcal{C}_2^{\rm far}} \sum_{\nu\in I_m} \sum_{\mu\in I_m} \kappa_{\nu,\mu}^b \, J_\tau^\nu(b_i) \, J_\sigma^\mu \end{aligned}$$

with quantities $J_\sigma^\mu(b_j)$ and J_σ^μ already defined in §2.8.

4 Hierarchical Matrices

The panel clustering method was element-oriented and enabled a fast matrix-vector multiplication. The present method is index-oriented and supports all matrix operations, i.e., additionally an approximate addition, multiplication and inversion of matrices are possible.

The technique of hierarchical matrices ($\mathcal{H}$-matrices) applies not only to full BEM matrices, but also to the fully populated inverse stiffness matrices of FEM problems.

Again the construction is based on trees which are similar to those from §3. However, the panels are replaced by the indices, i.e., by the degrees of freedom. This will lead to a block-structured matrix, where all subblocks are filled with low-rank matrices.

The use of low-rank matrices for subblocks was already proposed by [19], however, the construction of the efficient block-structure was missing.

4.1 Index Set I

We consider square matrices $\mathbf{A} = (a_{ij})_{i,j\in I}$, where the indices i, j run through the index set I of size $n := \#I$. We shall not use an explicit naming of the indices by $I = \{1, \ldots, n\}$, since this might lead to the wrong impression that the indices must have a special ordering. The technique of $\mathcal{H}$-matrices can easily be extended to rectangular matrices $\mathbf{B} = (b_{i,j})_{i\in I, j\in J}$, where I and J are different index sets.

For the following construction we need to know some geometric information about the indices. The simplest case is given by point data:

Assumption 4.1 *Each index $i \in I$ is associated with a 'nodal point' $\xi^i \in \mathbb{R}^d$.*

In this case, we use the following obvious definitions for the diameter of a subset $I' \subset I$ and for the distance of two subsets $I', I'' \subset I$:

$$\begin{aligned} \operatorname{diam}(I') &:= \max\left\{ \left|\xi^i - \xi^j\right| : i, j \in I' \right\}, \\ \operatorname{dist}(I', I'') &:= \min\left\{ \left|\xi^i - \xi^j\right| : i \in I', j \in I'' \right\}, \end{aligned} \tag{4.1a}$$

where $|\cdot|$ denotes the Euclidean norm in $\mathbb{R}^d$.

Although this information is sufficient for the practical application, precise statements (and proofs) about the FEM (or BEM) Galerkin method require the following support information:

Assumption 4.2 *Each index $i \in I$ is associated with the support $X_i := \operatorname{supp}(b_i) \subset \mathbb{R}^d$ of the finite element basis function b_i. For subsets $I', I'' \subset I$ we define*

$$\begin{aligned} X(I') &:= \textstyle\bigcup_{i\in I'} X_i, \\ \operatorname{diam}(I') &:= \max\{|\mathbf{x} - \mathbf{y}| : \mathbf{x}, \mathbf{y} \in X(I')\}, \\ \operatorname{dist}(I', I'') &:= \min\{|\mathbf{x} - \mathbf{y}| : \mathbf{x} \in X(I'),\ \mathbf{y} \in X(I'')\}. \end{aligned} \tag{4.1b}$$

4.2 Cluster Tree T_I for $\mathcal{H}$-Matrices

The following tree T_I is constructed as the panel cluster tree T from Definition 2.1, but the panels $t \in \mathcal{P}$ are replaced by indices $i \in I$.

Definition 4.3 *The cluster tree T_I consisting of subsets of I is structured as follows.*
(a) $I \in T_I$ is the root *of T_I.*
(b) The leaves *of T_I are given by the one-element sets $\{i\}$ for all $i \in I$.*
(c) If $\tau \in T_I$ is no leaf, there exist disjoint sons $\tau_1, \ldots, \tau_k \in T_I$ ($k = k(\tau) > 1$) with $\tau = \tau_1 \cup \ldots \cup \tau_k$.

We denote the set of sons by $S_I(\tau)$. Usually, binary trees ($k = 2$) are appropriate.

Remark 4.4 *(a) The fact that the leaves of T_I contain exactly one element is assumed in order to simplify the considerations. In practice, one fixes a number C_{leaf} (e.g., $C_{\mathrm{leaf}} = 32$) and deletes all $\tau' \in S_I(\tau)$ with $\#\tau \le C_{\mathrm{leaf}}$, i.e., in the reduced tree the leaves are characterised by $\#\tau \le C_{\mathrm{leaf}}$. Definition 4.3 corresponds to $C_{\mathrm{leaf}} = 1$.*

(b) The construction from §2.7.1 can be used as well to build T_I. The centres $\mathbf{z}_t$ from §2.7.1 (cf. Figure 2.2) are to be replaced by the points ξ^i from Assumption 4.1.

4.3 Block Cluster Tree $T_{I\times I}$

The entries a_{ij} of a matrix $\mathbf{A} \in \mathbb{R}^{I\times I}$ are indexed by pairs $(i,j) \in I \times I$. Accordingly, the block cluster tree $T_{I\times I}$ contains subsets of $I \times I$. Given the tree T_I, the block cluster tree $T_{I\times I}$ is constructed similarly to §3.1.1. The vertices (blocks) of $T_{I\times I}$ are denoted by b, the sons of b form the set $S_{I\times I}(b)$. For a matrix $\mathbf{A} \in \mathbb{R}^{I\times I}$ and a block $b \in T_{I\times I}$, the corresponding *submatrix* is denoted by

$$\mathbf{A}|_b := (a_{i,j})_{(i,j)\in b}\,. \tag{4.2}$$

Definition 4.5 *(a) The vertices of $T_{I\times I}$ are products $b = \tau \times \sigma$ of two clusters $\tau, \sigma \in T_I$.*
(b) $I \times I \in T_{I\times I}$ is the root of the tree.
(c) The set of sons of $b = \tau \times \sigma \in T_{I\times I}$ is defined by

$$S_{I\times I}(b) := \{\tau' \times \sigma' : \tau' \in S(\tau), \sigma' \in S(\sigma)\}.$$

Note that $S_{I\times I}(b) = \emptyset$ if either $S(\tau)$ or $S(\sigma)$ are the empty set. Hence, the leaves of $S_{I\times I}(b)$ are those $b = \tau \times \sigma$ where either τ or σ are leaves of T_I.

4.4 Admissibility Condition

Next, we need an *admissibility condition* that allows us to check if a block b is of appropriate size. We recall that $\operatorname{diam}(\sigma)$ and $\operatorname{dist}(\tau,\sigma)$ ($\tau, \sigma \in T_I$) are defined by (4.1a or b).

Definition 4.6 *Let $\eta > 0$ be a fixed parameter. The block $b = \tau \times \sigma \in T_{I\times I}$ is called* admissible*, if either b is a leaf or*

$$\min\{\operatorname{diam}(\tau), \operatorname{diam}(\sigma)\} \le \eta \operatorname{dist}(\tau,\sigma). \tag{4.3}$$

Note that in (4.3) the *minimum* of the diameters appears, while the panel clustering in (3.1) needs the maximum.

The simplest way to check the admissibility condition (4.3), is to apply (4.3) to the bounding boxes Q_σ and Q_τ from (2.22). The condition $\min\{\operatorname{diam}(Q_\sigma), \operatorname{diam}(Q_\tau)\} \le 2\eta \operatorname{dist}(Q_\sigma, Q_\tau)$ which is easy to verify, implies (4.3).

4.5 Admissible Block Partitioning

The first step in the construction of $\mathcal{H}$-matrices is the block partitioning (see, e.g., Figure 4.1). The partitioning called P is a covering in the sense that all blocks $b \in P$ are disjoint and $\bigcup_{b\in P} b = I \times I$. The partitioning is admissible if all $b \in P$ are admissible is the sense of Definition 4.6. Again, we are looking for a minimum admissible partitioning for which $\#P$ is as small as possible. It can be determined as in (2.8a,b) or (3.2a,b). We apply (4.4a) with $DivideP$ from (4.4b),

$$P := \emptyset;\ DivideP(I \times I, P); \tag{4.4a}$$

$$\begin{aligned}
&\textbf{procedure } DivideP(b, P);\ \textbf{comment } b \in T_{I\times I},\ P \subset T_{I\times I};\\
&\textbf{begin if } b \text{ is admissible } \textbf{then } P := P \cup \{b\}\\
&\qquad \textbf{else if } b \text{ is a leaf of } T_{I\times I} \textbf{ then } P := P \cup \{b\}\\
&\qquad\qquad \textbf{else for all } b' \in S_{I\times I}(b) \textbf{ do } DivideP(b', P)\\
&\textbf{end};
\end{aligned} \tag{4.4b}$$

Next, we give an example of such an minimum admissible partitioning which corresponds to a discretisation of the integral operator $\int_0^1 \log|x-y|\, f(y)dy$, where $d = 1$ is the spatial dimension. Consider the piecewise constant boundary elements which give rise to the supports

$$X_i := [(i-1)\,h, ih] \quad \text{for } i \in I := \{1, \dots, n\} \qquad \text{and } h := 1/n, \text{ where } n = 2^p$$

(cf. (4.1b)). The cluster tree T_I is the binary tree obtained by a uniform halving: The resulting clusters form the tree $T_I = \{\tau_i^\ell : 0 \le \ell \le p,\ 1 \le i \le 2^\ell\}$, where

$$\tau_i^\ell = \left\{(i-1) * 2^{p-\ell} + 1, (i-1) * 2^{p-\ell} + 2, \dots, i * 2^{p-\ell}\right\}. \tag{4.5}$$

Note that $\tau_1^0 = I$ is the root, while $\tau_i^p = \{i\}$ are the leaves. Further, we choose $\eta = 1$ in (4.3). Then the resulting block partitioning is shown in Figure 4.1. Under natural conditions, the number of blocks is $\#P = \mathcal{O}(n)$.

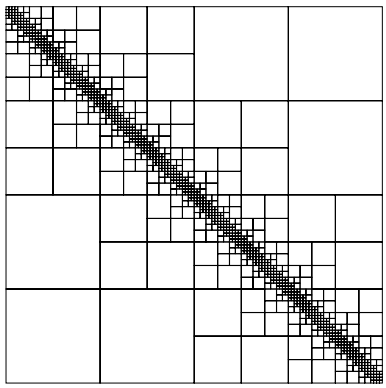

Figure 4.1: Partitioning for a 1D example

4.6 $\mathcal{H}$-Matrices and Rk-Matrices

Definition 4.7 ($\mathcal{H}$-matrix) *Given a cluster tree T_I for an index set I, let the related minimum admissible partitioning be denoted by P. Further, let $k \in \mathbb{N}$ be a given integer. Then the set $\mathcal{H}(k, P)$ consists of all matrices $\mathbf{M} \in \mathbb{R}^{I\times I}$ with*

$$\operatorname{rank}(\mathbf{M}|_b) \le k \qquad \textit{for all } b \in P.$$

Any rectangular matrix $\mathbf{C} \in \mathbb{R}^b$ ($b = \tau \times \sigma$) with $\operatorname{rank}(\mathbf{C}) \le k$ gives rise to the following equivalent representations

$$\begin{aligned}
&\mathbf{C} = \textstyle\sum_{i=1}^k \mathbf{a}_i \mathbf{b}_i^\top \in \mathbb{R}^b && \text{with vectors } \mathbf{a}_i \in \mathbb{R}^\sigma,\ \mathbf{b}_i \in \mathbb{R}^\tau,\\
&\mathbf{C} = \mathbf{A}\mathbf{B}^\top && \text{with } \mathbf{A} \in \mathbb{R}^{\tau\times k},\ \mathbf{B} \in \mathbb{R}^{\sigma\times k},
\end{aligned} \tag{4.6}$$

where the matrices $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_k]$, $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k]$ are composed by the vectors $\mathbf{a}_i, \mathbf{b}_i$. The vectors in (4.6) may be linearly dependent, since $\operatorname{rank}(\mathbf{M}) < k$ is not excluded. Throughout this section the bound k on the rank is assumed to be much smaller than the dimension n.

Definition 4.8 (Rk-matrix) *Matrices represented in the form (4.6) are called Rk-matrices.*

Remark 4.9 *(a) Rk-matrices require a storage of size* $2k\,(\#\sigma + \#\tau)$.

(b) Multiplication of an Rk-matrix $\mathbf{C} = \mathbf{A}\mathbf{B}^\top$ *with a vector requires* k *scalar products and vector additions:* $\mathbf{C}\mathbf{v} = \sum_{i=1}^{k} \alpha_i \mathbf{a}_i$ *with* $\alpha_i := \langle \mathbf{b}_i, \mathbf{v} \rangle$. *The cost is* $2k\,(\#\tau + \#\sigma)$.

(c) Multiplication of two Rk-matrices $\mathbf{R} = \mathbf{A}\mathbf{B}^\top \in \mathbb{R}^b, \mathbf{S} = \mathbf{C}\mathbf{D}^\top \in \mathbb{R}^{b'}$ *with* $b = \tau \times \sigma$, $b' = \sigma \times \sigma'$ *leads to the Rk-matrix* $\mathbf{T} = \mathbf{E}\mathbf{D}^\top \in \mathbb{R}^{\tau \times \sigma'}$ *with* $\mathbf{E} = \mathbf{A} * \mathbf{Z}$, *where* $\mathbf{Z} = \mathbf{B}^\top \mathbf{C}$ *is of size* $k \times k$. *The operation cost is* $2k^2\,(\#\tau + \#\sigma)$.

(d) The product $\mathbf{M}\mathbf{R}$ *for an arbitrary matrix* $\mathbf{M} \in \mathbb{R}^b$ *and an Rk-matrix* $\mathbf{R} \in \mathbb{R}^{b'}$ *is again an Rk-matrix of the form (4.6) with* $\mathbf{a}_i' := \mathbf{M}\mathbf{a}_i$.

According to Remark 4.4, the leaves of T_I may be characterised by $\#\tau \le C_{\mathrm{leaf}}$. Then all submatrices $\mathbf{M}|_b$ of an $\mathcal{H}$-matrix $\mathbf{M}$ are represented as Rk-matrices except for the case when $b = \tau \times \sigma$ with $\#\sigma, \#\tau \le C_{\mathrm{leaf}}$, where a (usual) full matrix is preferable.

Remark 4.10 *Under rather general assumptions on the tree* T_I *and on the geometric data* ξ^i *or* X_i *(cf. §4.1), the storage requirements for any* $\mathbf{M} \in \mathcal{H}(k, P)$ *are* $\mathcal{O}(nk \log(n))$ *(cf. [14], [15]).*

The constant in the estimate $\mathcal{O}(nk \log(n))$ from Remark 4.10 is determined by a *sparsity constant* C_{sp} of the partitioning P (see [9]).

4.7 Hierarchical Format for BEM and FEM Matrices

So far, the set $\mathcal{H}(k, P)$ of matrices is defined. It is still to be shown that matrices of this format are able to approximate well those matrices which we want to represent.

4.7.1 BEM Matrices

The second version of the panel clustering method is already quite close to the present form. In the former case, the data associated to a block $b = \tau \times \sigma \in T_2$ (notation in the sense of §3.1.1) describe the part $\int_\tau \int_\sigma \kappa(\mathbf{x}, \mathbf{y}) b_i(\mathbf{x}) u(\mathbf{y}) d\mathbf{x} d\mathbf{y}$ with κ approximated by $\tilde{\kappa}$. In the special case $u = b_j$ (i.e., $\mathbf{u}$ is the jth unit vector) this integral becomes $\int_\tau \int_\sigma \kappa(\mathbf{x}, \mathbf{y}) b_i(\mathbf{x}) b_j(\mathbf{y}) d\mathbf{x} d\mathbf{y}$. Now, we want to represent $a_{ij} = \int_\Gamma \int_\Gamma \kappa(\mathbf{x}, \mathbf{y}) b_i(\mathbf{x}) b_j(\mathbf{y}) d\mathbf{x} d\mathbf{y}$ which, in general, is different from the previous result, since the supports X_i, X_j of b_i, b_j are not necessarily contained in τ and σ. Due to the admissibility of the block $b = \tau \times \sigma \in P$ (notation in the sense of §4.2), $\kappa(\mathbf{x}, \mathbf{y})$ is approximated by

$$\tilde{\kappa}(\mathbf{x}, \mathbf{y}) = \sum_{\iota=1}^{k} \Psi_\iota(\mathbf{x}) \Phi_\iota(\mathbf{y}) \qquad \text{for all } \mathbf{x} \in X(\tau), \mathbf{y} \in X(\sigma) \tag{4.7}$$

(cf. (2.2)). Inserting one term of (4.7) into $\int_\Gamma \int_\Gamma \ldots b_i(\mathbf{x}) b_j(\mathbf{y}) d\mathbf{x} d\mathbf{y}$ for $(i, j) \in b$, we obtain $\alpha_i * \beta_j$, where $\alpha_i := \int_\Gamma \Psi_\iota(\mathbf{x}) b_i(\mathbf{x}) d\mathbf{x}$ and $\beta_j = \int_\Gamma \Phi_\iota(\mathbf{y}) b_j(\mathbf{y}) d\mathbf{y}$. These components form the vectors $\mathbf{a}_\iota = (\alpha_i)_{i \in \tau}$ and $\mathbf{b}_\iota = (\beta_j)_{j \in \sigma}$. Hence,

$$\int_\Gamma \int_\Gamma \tilde{\kappa}(\mathbf{x}, \mathbf{y}) b_i(\mathbf{x}) b_j(\mathbf{y}) d\mathbf{x} d\mathbf{y} = \sum_{\iota=1}^{k} \mathbf{a}_\iota \mathbf{b}_\iota^\top \qquad \text{for } (i, j) \in b$$

shows that the approximation of κ in $X(\tau) \times X(\sigma)$ by $\tilde{\kappa}$ containing k terms is equivalent to having an Rk-submatrix $\mathbf{A}|_b$ with $\operatorname{rank}(\mathbf{A}|_b) \le k$. In the case of (admissible) blocks $b \in P$, which do not satisfy (4.3), $b = \{(i, j)\}$ is of size 1×1 (cf. Definition 4.6), so that a_{ij} can be defined by the exact entry.

Remark 4.11 *(a) Let $\mathbf{A} \in \mathbb{R}^{I\times I}$ be the exact BEM matrix. The existence of a (well approximating) $\mathcal{H}$-matrix $\tilde{\mathbf{A}} \in \mathcal{H}(k,P)$ follows from a sufficiently accurate expansion (4.7) with k terms for all 'far-field blocks' $b \in P$ satisfying (4.3).*

(b) The BEM kernels (mathematically more precisely: asymptotically smooth kernels, cf. [15]) allow an approximation up to an error of $\mathcal{O}(\eta^{\sqrt[d-1]{k}})$ by k terms (η is the factor in (4.3)).

Concerning the construction of $\tilde{\mathbf{A}} \in \mathcal{H}(k,P)$ one can follow the pattern of panel clustering (see, e.g., [5] and [4]). Interestingly, there is another approach (called "adaptive cross approximation" (ACA)) by [1], [3], which only makes use of the procedure $(i,j) \mapsto \int_\Gamma \int_\Gamma \kappa(\mathbf{x},\mathbf{y}) b_i(\mathbf{x}) b_j(\mathbf{y}) d\mathbf{x} d\mathbf{y}$ (this mapping is evaluated only for a few index pairs $(i,j) \in I \times I$).

4.7.2 FEM Matrices

Since FEM matrices are sparse, we have the following trivial statement.

Remark 4.12 *Let (4.1b) be used to define the admissible partitioning P. Then, for any $k \geq 1$, a FEM stiffness matrix belongs to $\mathcal{H}(k,P)$.*

The reason is that $\mathbf{A}|_b = \mathbf{O}$ for all blocks b satisfying (4.3), since (4.3) implies that the supports of the basis functions b_i, b_j ($(i,j) \in b$) are disjoint. Remark 4.12 expresses that fact that $\mathbf{A}$ can be considered as $\mathcal{H}$-matrix. Therefore, we can immediately apply the matrix operations described below. In particular, the inverse matrix can be determined approximately. The latter task requires that $\mathbf{A}^{-1}$ has a good approximation $\mathbf{B} \in \mathcal{H}(k,P)$. This property is the subject of the next theorem.

Theorem 4.13 *Let $Lu = -\sum_{\nu,\mu=1}^{d} \partial_\mu (c_{\nu,\mu} \partial_\nu u)$ be a uniformly elliptic differential operator whose coefficients are allowed to be extremely non-smooth: $c_{ij} \in L^\infty(\Omega)$. Let $\mathbf{A}$ be a FEM stiffness matrix for this boundary value problem. Then there are approximants $\mathbf{B}_k \in \mathcal{H}(k,P)$ so that $\mathbf{B}_k$ converges exponentially to $\mathbf{A}^{-1}$ (details in [2]).*

4.8 Matrix Operations

In the following, we describe the matrix operations which can be performed using $\mathcal{H}$-matrices. Except the matrix-vector multiplication, the operations are approximate ones, but the accuracy can be controlled by means of the rank parameter k. Concerning further details and cost estimates, we refer to [9].

4.8.1 Matrix-Vector Multiplication

The matrix-vector product $\mathbf{y} \mapsto \mathbf{y} + \mathbf{M}\mathbf{x}$ is performed by the call $MVM(\mathbf{M}, I\times I, \mathbf{x}, \mathbf{y})$ of

```
procedure MVM(M, b, x, y); comment b = τ × σ ∈ T_{I×I}, M ∈ R^{I×I}, x, y ∈ R^I;
begin if S_{I×I}(b) ≠ ∅ then for b' ∈ S_{I×I}(b) do MVM(M, b', x, y)
        else y|_τ := y|_τ + M|_b x|_σ
end;
```
(4.8)

The third line of (4.8) uses the matrix-vector multiplication of an Rk-matrix with a vector (see Remark 4.9b). The overall arithmetical cost is $\mathcal{O}(nk \log n)$.

4.8.2 Matrix-Matrix Addition, Truncation

For $\mathbf{M}', \mathbf{M}'' \in \mathcal{H}(k,P)$, the exact sum $\mathbf{M} := \mathbf{M}' + \mathbf{M}''$ is obtained by summing $\mathbf{M}'|_b + \mathbf{M}''|_b$ over all blocks $b \in P$. The problem is, however, that usually $\mathbf{M}'|_b + \mathbf{M}''|_b$ has rank $2k$, so that a fill-in occurs and $\mathbf{M}$ is no longer in the set $\mathcal{H}(k,P)$. Therefore, a truncation of $\mathbf{M}|_b = \mathbf{M}'|_b + \mathbf{M}''|_b$ back to an Rk-matrix $\tilde{\mathbf{M}}|_b$ is applied.

Concerning the truncation, we recall the optimal approximation of a general (rectangular) matrix $\mathbf{M} \in \mathbb{R}^{\tau\times\sigma}$ by an Rk-matrix $\tilde{\mathbf{M}}$. Optimality holds with respect to the spectral norm ($\|\mathbf{A}\| = \max\{|\mathbf{A}\mathbf{x}| \,/\, |\mathbf{x}| : \mathbf{x} \neq \mathbf{0}\} = \sqrt{\lambda_{\max}}$, where $\lambda_{\max}$ is the maximum eigenvalue of $\mathbf{A}\mathbf{A}^\top$) and the Frobenius norm ($\|\mathbf{A}\|_F = (\sum_{i,j} a_{i,j}^2)^{1/2}$).

Algorithm 4.14 *(a) Calculate the* singular value decomposition $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ *of* $\mathbf{M}$*, i.e.,* $\mathbf{U}, \mathbf{V}$ *are unitary matrices, while* $\mathbf{\Sigma} = \operatorname{diag}(\sigma_1, \ldots)$ *is a diagonal rectangular matrix containing the singular values* $\sigma_1 \geq \sigma_2 \geq \ldots$.

(b) Set $\tilde{\mathbf{U}} := [\mathbf{U}_1, \ldots, \mathbf{U}_k]$ *(first* k *columns of* $\mathbf{U}$*),* $\tilde{\mathbf{\Sigma}} := \operatorname{diag}(\sigma_1, \ldots, \sigma_k)$ *(first (largest)* k *singular values),* $\tilde{\mathbf{V}} := [\mathbf{V}_1, \ldots, \mathbf{V}_k]$ *(first* k *columns of* $\mathbf{V}$*).*

(c) Set $\tilde{\mathbf{A}} := \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}} \in \mathbb{R}^{\tau \times k}$ *and* $\tilde{\mathbf{B}} := \tilde{\mathbf{V}} \in \mathbb{R}^{\sigma \times k}$ *in (4.6). Then* $\tilde{\mathbf{M}} = \tilde{\mathbf{A}}\tilde{\mathbf{B}}^\top$ *is the best Rk-matrix approximation of* $\mathbf{M}$*.*

We call $\tilde{\mathbf{M}}$ a truncation of $\mathbf{M}$ to the set of Rk-matrices. The costs are in general $\mathcal{O}((\#\tau + \#\sigma)^3)$ operations. In our application, the sum $\mathbf{M} := \mathbf{M}' + \mathbf{M}''$ has rank $K \leq 2k$. Here we can apply a cheaper singular value decomposition.

Algorithm 4.15 *Let* $\mathbf{M} = \mathbf{A}\mathbf{B}^\top$ *be an RK-matrix with* $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{\tau \times K}$ *and* $K > k$*.*

(a) Calculate a truncated QR-decomposition $\mathbf{A} = \mathbf{Q}_A\mathbf{R}_A$ *of* $\mathbf{A}$*, i.e.,* $\mathbf{Q}_A \in \mathbb{R}^{\tau \times K}$*,* $\mathbf{Q}_A^\top\mathbf{Q}_A = \mathbf{I}$*, and* $\mathbf{R}_A \in \mathbb{R}^{K \times K}$ *upper triangular matrix.*

(b) Calculate a truncated QR-decomposition $\mathbf{B} = \mathbf{Q}_B\mathbf{R}_B$ *of* $\mathbf{B}$*,* $\mathbf{Q}_B \in \mathbb{R}^{\sigma \times K}$*,* $\mathbf{R}_B \in \mathbb{R}^{K \times K}$*.*

(c) Calculate a singular value decomposition $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ *of the* $K \times K$ *matrix* $\mathbf{R}_A\mathbf{R}_B^\top$*.*

(d) Set $\tilde{\mathbf{U}}$*,* $\tilde{\mathbf{\Sigma}}$*,,* $\tilde{\mathbf{V}}$ *as in Algorithm 4.14b.*

(e) Set $\tilde{\mathbf{A}} := \mathbf{Q}_A\tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}} \in \mathbb{R}^{\tau \times k}$ *and* $\tilde{\mathbf{B}} := \mathbf{Q}_B\tilde{\mathbf{V}} \in \mathbb{R}^{\sigma \times k}$*. Then,* $\tilde{\mathbf{M}} = \tilde{\mathbf{A}}\tilde{\mathbf{B}}^\top$ *is the best Rk-matrix approximation of* $\mathbf{M}$*.*

The truncation from above costs $\mathcal{O}(K^2(\#\tau + \#\sigma) + K^3)$ arithmetical operations.

The exact addition $\mathbf{M}', \mathbf{M}'' \in \mathcal{H}(k, P) \mapsto \mathbf{M} := \mathbf{M}' + \mathbf{M}'' \in \mathcal{H}(2k, P)$ together with the truncation $\mathbf{M} \in \mathcal{H}(2k, P) \mapsto \tilde{\mathbf{M}} \in \mathcal{H}(k, P)$ is denoted by the *formatted addition*

$$\mathbf{M}' \oplus \mathbf{M}''. \tag{4.9}$$

Similarly, the *formatted subtraction* $\ominus$ is defined. The complexity of $\oplus$ and $\ominus$ is $\mathcal{O}(nk^2 \log n)$.

4.8.3 Matrix-Matrix Multiplication

Let $\mathbf{X}, \mathbf{Y} \in \mathcal{H}(k, P)$. Under the assumption that T_I is a binary tree, both matrices are substructured by $\mathbf{X} = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}$, $\mathbf{Y} = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix}$, and the product is

$$\mathbf{X}\mathbf{Y} = \begin{bmatrix} X_{11}Y_{11} + X_{12}Y_{21} & X_{11}Y_{12} + X_{12}Y_{22} \\ X_{21}Y_{11} + X_{22}Y_{21} & X_{21}Y_{12} + X_{22}Y_{22} \end{bmatrix}.$$

The most costly subproducts are $X_{11}Y_{11}$ and $X_{22}Y_{22}$, since these submatrices have the finest partitioning, whereas $X_{12}, Y_{21}, X_{21}, Y_{12}$ have a coarser format. Performing the products recursively and adding according to §4.8.2, we obtain an approximate multiplication $\mathbf{X} \odot \mathbf{Y}$. Its costs are $\mathcal{O}(nk^2 \log^2 n)$ (cf. [14]). A detailed algorithm can be found in [9] and [4].

4.8.4 Inversion

Let $\mathbf{A} \in \mathcal{H}(k, P)$. Under the assumption that T_I is a binary tree, we have as above that $\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$. The inverse of a 2×2 block-matrix can be computed by the block-Gauss elimination (see [14]) if the principal submatrices are invertible:

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{A}_{11}^{-1} + \mathbf{A}_{11}^{-1}\mathbf{A}_{12}\mathbf{S}^{-1}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} & -\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\mathbf{S}^{-1} \\ -\mathbf{S}^{-1}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} & \mathbf{S}^{-1} \end{bmatrix} \quad \begin{array}{l} \text{with} \\ \mathbf{S} = \mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12}. \end{array} \tag{4.10}$$

Applying a recursive procedure Inv, compute $Inv(\mathbf{A}_{11})$ as approximation of $\mathbf{A}_{11}^{-1}$, invert $\tilde{\mathbf{S}} := \mathbf{A}_{22} \ominus \mathbf{A}_{21} \odot Inv(\mathbf{A}_{11}) \odot \mathbf{A}_{12}$ and perform the remaining operations in (4.10) by means of $\oplus$ and $\odot$. Again, the precise algorithm is in [4].

The complexity for the computation of the formatted inverse is $\mathcal{O}(nk^2 \log^2 n)$ (cf. [14], [8], [9]).

4.9 Examples

4.9.1 BEM Case

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 using a Galerkin method with piecewise constant basis functions. The logarithmic kernel function is approximated by the interpolatory approach from §2.7.2 (interpolation at Chebyshev points).

n	1	2	3	4	5
1024	3.57_{10}-2	2.16_{10}-3	2.50_{10}-4	7.88_{10}-6	2.67_{10}-6
2048	3.58_{10}-2	2.19_{10}-3	2.51_{10}-4	7.86_{10}-6	2.69_{10}-6
4096	3.59_{10}-2	2.20_{10}-3	2.51_{10}-4	7.87_{10}-6	2.68_{10}-6
8192	3.59_{10}-2	2.20_{10}-3	2.52_{10}-4	7.76_{10}-6	2.67_{10}-6
16384	3.59_{10}-2	2.21_{10}-3	2.53_{10}-4	7.87_{10}-6	2.68_{10}-6

Table 4.1: Approximation error for the single layer potential

The first column of Table 4.1 contains the number of degrees of freedom ($n = \#I$), the following columns give the relative error $\|\mathbf{A} - \tilde{\mathbf{A}}\|/\|\mathbf{A}\|$ (spectral norm). 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	0.01	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 4.2: Time [sec] required for the matrix-vector multiplication (single layer potential)

n	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 4.3: Time [sec] required for building the $\mathcal{H}$-matrix (single layer potential)

The time (SUN Enterprise 6000 using 248 MHz UltraSPARC II) required for matrix vector multiplications is given in Table 4.2. We can see that the complexity grows almost linearly in the number of degrees of freedom and rather slowly with respect to the interpolation order.

Finally, we consider the time required for building the $\mathcal{H}$-matrix representation of the discretised integral operator (see Table 4.3). The integral of the Lagrange polynomials is computed by using an exact Gauss quadrature formula, while the integral of the kernel function is computed analytically. Once more we observe

k	$n=4096$	16384	65536	262144
1	2.4	8.9	2.6+1	4.7+1
2	5.7-1	3.2	1.2+1	2.7+1
3	9.2-2	5.2-1	2.4	1.0+1
4	2.0-2	9.9-2	4.4-1	1.91
5	2.3-3	9.2-3	4.0-2	1.7-1
6	6.4-4	3.7-3	1.8-2	8.4-2
7	1.4-4	6.9-4	2.9-3	1.2-2
8	7.8-5	3.9-4	1.8-3	7.7-3
9	8.5-6	4.6-5	2.1-4	9.4-4
15	6.8-9	3.3-8	1.3-7	5.2-7
20	1.7-12	1.3-10	5.3-10	2.5-9

k	$n=6664$	13568	27384	55024	110312
1	9.6-2	9.9-2	7.9-2	1.1-1	9.4-2
2	1.3-2	1.1-2	1.7-2	1.9-2	1.6-2
3	3.9-3	4.4-3	1.7-3	4.5-3	4.7-3
4	8.6-5	4.7-4	1.7-4	5.0-4	5.1-4
5	8.9-6	3.6-5	7.6-6	4.9-5	5.0-5
6	2.1-8	9.8-7	1.2-6	1.3-6	1.4-6
7	3.1-10	5.0-7	1.9-10	5.8-7	5.9-7
8	1.4-12	4.2-10	2.1-11	2.5-10	2.8-10
9	1.0-14	2.4-13	2.1-14	2.7-13	2.8-13

Table 4.4: Relative error $\|\mathbf{I} - \mathbf{A}\,\widetilde{\mathbf{A}^{-1}}\|$ in the spectral norm for the (formatted) inverse on a uniform grid (left) and on the boundary concentrated mesh (right).

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 the specified rather slow processor, the boundary element matrix for more than half a million degrees of freedom can be approximated with an error $< 0.03\%$ in less than half an hour.

4.9.2 FEM Case, Inverse Stiffness Matrix

We give a short summary of numerical tests from [9] and consider first the Poisson equation $-\Delta u = f$ on the unit square $\Omega = [0,1]^2$ with zero boundary condition $u = 0$ on $\Gamma = \partial\Omega$. The approximate inverse $\widetilde{\mathbf{A}^{-1}}$ is computed for different local ranks k. The left part of Table 4.4 shows the relative error $\|\mathbf{I} - \mathbf{A}\,\widetilde{\mathbf{A}^{-1}}\|$ in the spectral norm for the (formatted) inverse on a uniform grid.

Next we show that the uniformity of the grid and the simple shape of the square do not play any role. The grid from Figure 4.2 is strongly graded towards the boundary ("boundary concentrated mesh"). For the details of the geometrically balanced cluster tree we refer to [9]. The complexity of the inversion is reduced as compared to the uniform case while the accuracy is enhanced (see right part of Table 4.4). This resembles the fact that the grid mainly degenerates to a lower dimensional structure (the boundary).

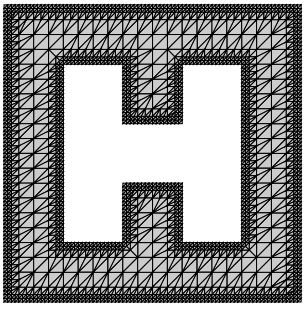

Figure 4.2: The boundary concentrated mesh

Finally, we give examples showing that the performance is not deteriorated by rough coefficients. Consider the differential equation

$$\begin{aligned} -\operatorname{div}(\sigma(x,y)\nabla u(x,y)) &= f(x,y) \qquad \text{in } \Omega = [0,1]^2, \\ u &= 0 \qquad\qquad\quad \text{on } \Gamma = \partial\Omega, \end{aligned} \tag{4.11}$$

Let $\Omega_1 \subset \Omega$ be the wall-like domain from Figure 4.3. Let L_a be the differential operator in (4.11) with

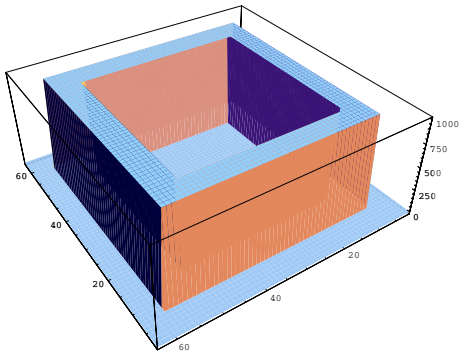

Figure 4.3: Subdomain Ω_1 of $\Omega = [0,1]^2$

$n = 2304$	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$
Storage (MB)	10.2	18.9	27.6	36.2		
Δ	4.1_{10}-03	5.9_{10}-04	1.1_{10}-05	1.2_{10}-06		
L_{10^3}	6.9_{10}-03	9.8_{10}-04	1.6_{10}-05	2.1_{10}-06		
L_{10^6}	6.9_{10}-03	9.8_{10}-04	1.6_{10}-05	1.7_{10}-06		
$n = 6400$						
Storage (MB)	40.0	75.9	111.6	147.5	183.1	218.8
Δ	3.5_{10}-03	6.5_{10}-04	8.8_{10}-06	2.1_{10}-06	4.2_{10}-07	8.3_{10}-09
L_{10^3}	5.5_{10}-03	1.0_{10}-03	1.2_{10}-05	3.2_{10}-06	5.5_{10}-08	1.3_{10}-08
L_{10^6}	5.6_{10}-03	1.0_{10}-03	1.2_{10}-05	3.1_{10}-07	4.7_{10}-08	9.1_{10}-09
$n = 14400$						
Storage (MB)	123.4	235.7	349.6	462.0	575.9	688.2
Δ	3.2_{10}-03	5.9_{10}-04	8.9_{10}-06	2.3_{10}-06	5.5_{10}-08	1.5_{10}-08
L_{10^3}	4.9_{10}-03	8.8_{10}-04	1.2_{10}-05	3.3_{10}-06	7.3_{10}-08	1.9_{10}-08
L_{10^6}	5.0_{10}-03	8.8_{10}-04	1.0_{10}-05	3.2_{10}-06	6.7_{10}-08	9.1_{10}-09

Table 4.5: Frobenius norm $\|\mathbf{A}^{-1} - \widetilde{\mathbf{A}^{-1}}\|$ of the best approximation to $\mathbf{A}^{-1}$ using the local rank k

$\sigma(x,y) = \left\{ \begin{array}{ll} a, & (x,y) \in \Omega_1 \\ 1, & (x,y) \in \Omega \backslash \Omega_1 \end{array} \right\}$. Note that $L_1 = -\Delta$. Table 4.5 shows the relative accuracy measured for different problem sizes n in Frobenius norm when approximating the inverse of the respective FEM matrix by an $\mathcal{H}$-matrix with local rank k. The results demonstrate that the error $\|\mathbf{A}^{-1} - \widetilde{\mathbf{A}^{-1}}\|$ depends on the jump a very weakly.

4.10 $\mathcal{H}^2$-Matrices and Other Variants of $\mathcal{H}$-Matrices

4.10.1 Variable Rank, Recompression

We may replace the integer k in Definition 4.7 by a function $k(b)$. Then, the matrix $\mathbf{M}$ has to fulfil $\operatorname{rank}(\mathbf{M}|_b) \leq k(b)$ for all $b \in P$. A possible non-constant choice is $k(b) := \alpha * l(b)$, where $l(\cdot)$ is defined by induction: $l(b) = 1$ for leaves $b \in T_{I\times I}$ and $l(b) = 1 + \min\{l(b') : b' \in S_{I\times I}(b)\}$. In this case, $k(b)$ varies between 1 and $\log_2 n$. The low ranks correspond to the (many) small blocks, whereas large ranks occur for the few large blocks. As a result, cost estimates by $\mathcal{O}(nk^s \log^q n)$ for fixed k may turn into the optimal order $\mathcal{O}(n)$ for appropriate variable rank.

Given an $\mathcal{H}$-matrix $\mathbf{M}$ with a certain (variable or constant) local rank, it might happen that the block matrices $\mathbf{M}|_b$ can be reduced to lower rank with almost the same accuracy. The standard tool is a singular value decomposition of $\mathbf{M}|_b$. If some of the $k(b)$ singular values $\sigma_1 \geq \ldots \geq \sigma_{k(b)}$ are sufficiently small, these contributions can be omitted resulting in a smaller local rank $k(b)$.

4.10.2 Uniform $\mathcal{H}$-Matrices

Consider $b = \tau \times \sigma \in T_{I\times I}$. The submatrix $\mathbf{M}|_b$ belongs to $\mathbb{R}^{\tau\times\sigma}$. A special subspace of $\mathbb{R}^{\tau\times\sigma}$ is the tensor product space $V_b \otimes W_b = \{\mathbf{v}\mathbf{w}^\top : \mathbf{v} \in V_b, \mathbf{w} \in W_b\}$ of $V_b \in \mathbb{R}^\tau$ and $W_b \in \mathbb{R}^\sigma$. Note that $\mathbf{T} \in V_b \otimes W_b$ implies $\operatorname{rank}\mathbf{T} \le \min\{\dim V_b, \dim W_b\}$. Hence, we may replace the condition $\operatorname{rank}(\mathbf{M}|_b) \le k$ by $\mathbf{M}|_b \in V_b \otimes W_b$ with spaces V_b, W_b of dimension $\le k$. The resulting subset of $\mathcal{H}$-matrices is called the set of *uniform* $\mathcal{H}$-matrices.

For the representation of submatrices $\mathbf{M}|_b$, one uses corresponding bases $\{\mathbf{v}_1, \dots, \mathbf{v}_{\dim V_b}\}$ and $\{\mathbf{w}_1, \dots, \mathbf{w}_{\dim W_b}\}$ of V_b, W_b and defines $\mathbf{V}_b := [\mathbf{v}_1, \dots, \mathbf{v}_{\dim V_b}]$, $\mathbf{W}_b = [\mathbf{w}_1, \dots, \mathbf{w}_{\dim W_b}]$. Then, $\mathbf{M}|_b = \mathbf{V}_b\mathbf{S}_b\mathbf{W}_b^\top$, where the matrix $\mathbf{S}_b$ of size $\dim V_b \times \dim W_b$ contains the specific data of $\mathbf{M}|_b$ which are to be stored.

4.10.3 $\mathcal{H}^2$-Matrices

The previous class of uniform $\mathcal{H}$-matrices uses different spaces V_b, W_b for every $b = \tau\times\sigma \in P$. Now, we require that V_b depends only on τ, while W_b depends only on σ. Hence, we may start from a family $V = (V_\tau)_{\tau\in T_I}$ of spaces $V_\tau \subset \mathbb{R}^\tau$ and require $\mathbf{M}|_b \in V_\tau \otimes V_\sigma$ for all $b = \tau \times \sigma \in P$.

The second, characteristic requirement is the consistency condition

$$V_\tau|_{\tau'} \subseteq V_{\tau'} \qquad \text{for all } \tau \in T_I \text{ and } \tau' \in S(\tau), \tag{4.12}$$

i.e., $\mathbf{v}|_{\tau'} \in V_{\tau'}$ for all $\mathbf{v} \in V_\tau$. Let $\mathbf{V}_\tau$ and $\mathbf{V}_{\tau'}$ the corresponding bases. Due to (4.12), there is a matrix $\mathbf{B}_{\tau',\tau}$ such that $\mathbf{V}_\tau|_{\tau'} = \mathbf{V}_{\tau'}\mathbf{B}_{\tau',\tau}$. Thanks to Definition 4.3c, $\mathbf{V}_\tau$ can be obtained from $\{\mathbf{V}_{\tau'}, \mathbf{B}_{\tau',\tau} : \tau' \in S(\tau)\}$. Hence, the bases $\mathbf{V}_\tau$ need not be stored, instead the transformation matrices $\mathbf{B}_{\tau',\tau}$ are stored. This is an advantage, since their size is $k_{\tau'} \times k_\tau$ with $k_\tau := \dim \mathbf{V}_\tau \le k$ independent of the size of the blocks $b \in P$.

For details about $\mathcal{H}^2$-matrices, we refer to [5], [16]. The latter paper considers, e.g., the combination of the $\mathcal{H}^2$-matrix structure with variable dimensions k_τ. In [4], the example from §4.9.1 is computed also by means of the $\mathcal{H}^2$-matrix technique and numbers corresponding to Tables 4.1-4.3 are given. They show that a slightly reduced accuracy is obtained with considerably less work.

4.11 Applications

We mention three different fields for the application of $\mathcal{H}$-matrices.

4.11.1 Direct Use

In the case of a BEM matrix $\mathbf{A}$, the storage of the n^2 matrix entries must be avoided. Then the approximation of $\mathbf{A}$ by an $\mathcal{H}$-matrix $\tilde{\mathbf{A}} \in \mathcal{H}(k, P)$ reduces the storage requirements to almost $\mathcal{O}(n)$. Since in this case, $\tilde{\mathbf{A}}$ must carry all information about the BEM problem, the rank k must be chosen high enough (e.g., $k = \mathcal{O}(\log n)$) in order to maintain the accuracy. Second, the matrix-vector multiplication can be performed with almost linear cost.

For the solution of the system of linear equations $\mathbf{A}\mathbf{x} = \mathbf{b}$ one has two options: (a) Use an iterative scheme which is based on the matrix-vector multiplication (cg-type methods, multi-grid). (b) Compute the approximate inverse $\widetilde{\mathbf{A}^{-1}}$ (see §4.8.4).

The computation of operators like Steklov operators (Neumann-to-Dirichlet or Dirichlet-to-Neumann map) needs to perform the matrix-matrix multiplication $\odot$ from §4.8.3.

4.11.2 Rough Inverse

In the FEM case, the problem data are given by the sparse stiffness matrix $\mathbf{A}$. The approximate inverse $\widetilde{\mathbf{A}^{-1}}$ must be accurate enough, if $\tilde{\mathbf{x}} := \widetilde{\mathbf{A}^{-1}}\mathbf{b}$ should be a good approximation of the solution $\mathbf{x}$. However, there is no need to have $\widetilde{\mathbf{A}^{-1}}$ very accurate. Instead, one can use $\widetilde{\mathbf{A}^{-1}}$ as 'preconditioner': The iteration

$$\mathbf{x}^{m+1} := \mathbf{x}^m - \widetilde{\mathbf{A}^{-1}}\,(\mathbf{A}\mathbf{x}^m - \mathbf{b})$$

can be applied to improve $\mathbf{x}^0 := \widetilde{\mathbf{A}^{-1}}\mathbf{b}$. The convergence rate is given by the spectral radius of $\mathbf{I} - \widetilde{\mathbf{A}^{-1}}\mathbf{A}$ (cf. [12]). An upper bound of the spectral radius is the norm $\|\mathbf{I} - \widetilde{\mathbf{A}^{-1}}\mathbf{A}\|$ which should be < 1. For the elliptic example, this norm is given in Table 4.4.

4.11.3 Matrix-Valued Problems

There are further problem, where the usual matrix-vector approach is insufficient, since one is interested in matrices instead of vectors. We give some examples, which can be solved by means of the $\mathcal{H}$-matrix technique.

Matrix Exponential Function Matrix functions like the matrix exponential can be computed effectively by use of the Dunford-Cauchy representation

$$\exp(\mathbf{A}) = \frac{1}{2\pi i}\int_\Gamma \exp(z)(z\mathbf{I}-\mathbf{A})^{-1}dz, \tag{4.13}$$

where Γ is a curve in the complex plane containing the spectrum of $\mathbf{A}$ in its interior. Approximation of the integral by a quadrature rule (z_ν: quadrature points) leads to

$$\exp_{\mathcal{H}}(\mathbf{A}) = \sum_{\nu=-N}^{N} e^{-tz_\nu} a_\nu (z_\nu \mathbf{I}-\mathbf{A})^{-1}. \tag{4.14}$$

Since the integration error decreases exponentially with respect to N, one may choose $N = \mathcal{O}(\log^{3/2}\frac{1}{\varepsilon})$ to obtain an integration error ε. The resolvents $(z_\nu\mathbf{I}-\mathbf{A})^{-1}$ are computed due to §4.8.4. For further details we refer to [7].

Lyapunov Equation There are linear equations for matrices. An example is the Lyapunov equation $\mathbf{AX}+\mathbf{XB}+\mathbf{C}=\mathbf{O}$ for the unknown matrix $\mathbf{X}$, while $\mathbf{A},\mathbf{B},\mathbf{C}$ are given. One possible solution uses the representation $\mathbf{X} = \int_0^\infty e^{t\mathbf{A}}\mathbf{C}e^{t\mathbf{B}}dt$, provided that the eigenvalues of $\mathbf{A},\mathbf{B}$ have negative real parts. Since the dependence of $\exp_{\mathcal{H}}(\mathbf{A})$ on t in (4.14) is expressed by the scalar factor e^{-tz_ν}, one can replace $e^{t\mathbf{A}}, e^{t\mathbf{B}}$ by $\exp_{\mathcal{H}}(\mathbf{A})$ and $\exp_{\mathcal{H}}(\mathbf{B})$ and perform the integration exactly (cf. [7, Section 4.2]).

Riccati Equation For optimal control problems, the (nonlinear) Riccati equation

$$\mathbf{A}^\top\mathbf{X}+\mathbf{XA}-\mathbf{XFX}+\mathbf{G}=\mathbf{O} \qquad (\mathbf{A},\mathbf{F},\mathbf{G} \text{ given matrices, } \mathbf{X} \text{ to be determined})$$

is of interest. In [10] the direct representation of $\mathbf{X}$ by means of the matrix-valued sign function is applied. Its iterative computation requires again the inversion, which is provided by the $\mathcal{H}$-matrix technique.

References

[1] Bebendorf M. Approximation of boundary element matrices. *Numer. Math.* 2000; **86**:565–589.

[2] Bebendorf M and Hackbusch W. Existence of $\mathcal{H}$-matrix approximants to the inverse FE-matrix of elliptic operators with L^∞-coefficients. *Numer. Math.* 2003; ***95***:1-28.

[3] Bebendorf M and Rjasanov S. *Adaptive low-rank approximation of collocation matrices*, *Computing* 2003; **70**:1-24

[4] Börm S, Grasedyck L and Hackbusch W. Introduction to hierarchical matrices with applications. *Eng. Anal. Bound. Elem.*, 2003; **27**:405–422.

[5] Börm S and Hackbusch W. $\mathcal{H}^2$-matrix approximation of integral operators by interpolation, *Apl. Numer. Math.* 2002; **43**:129-143.

[6] Dahmen W, Prössdorf S and Schneider R. Wavelet approximation methods for pseudodifferential equations II: Matrix compression and fast solution. *Adv. Comput. Math.* 1993; **1**:259-335.

[7] Gavrilyuk I, Hackbusch W and Khoromskij BN. $\mathcal{H}$-matrix approximation for the operator exponential with applications. *Numer. Math.* 2002; **92**:83-111.

[8] Grasedyck L. *Theorie und Anwendungen Hierarchischer Matrizen.* Doctoral thesis. Universität Kiel, 2001.

[9] Grasedyck L and Hackbusch W. Construction and arithmetics of $\mathcal{H}$-matrices, *Computing* 2003; ***71*** (to appear).

[10] Grasedyck L, Hackbusch W and Khoromskij BN. Solution of large scale algebraic matrix Riccati equations by use of hierarchical matrices. *Computing* 2003; **70**:121-165

[11] Greengard L and Rokhlin V. A new version of the fast multipole method for the Laplace equation in three dimensions. *Acta Numerica* 1997; **6**:229-269.

[12] Hackbusch W. *Iterative Solution of Large Sparse Systems.* Springer: New York, 1994 — 2nd German edition: *Iterative Lösung großer schwachbesetzter Gleichungssysteme.* Teubner: Stuttgart, 1993.

[13] Hackbusch W. *Integral Equations. Theory and Numerical Treatment.* ISNM 128. Birkhäuser: Basel, 1995 — 2nd German edition: *Integralgleichungen. Theorie und Numerik.* Teubner: Stuttgart, 1997.

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

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

[16] Hackbusch W, Khoromskij BN and Sauter S. On $\mathcal{H}^2$-matrices. In *Lectures on Applied Mathematics*, Bungartz H, Hoppe R and Zenger C (eds). Springer: Heidelberg, 2000; 9–29.

[17] Hackbusch W and Nowak ZP. O cloznosti metoda panelej. In *Vycislitel'nye prozessy i sistemy,* Marchuk GI (ed). Nauka: Moscow, 1988; 233–244 (conference in Moscow, September 1986).

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

[19] Tyrtyshnikov E. Mosaic-skeleton approximation, *Calcolo* 1996; **33**:47–57.