Lecture Notes
in Computational Science
and Engineering

63

Editors

Mario Bebendorf

Hierarchical Matrices

A Means to Efficiently Solve Elliptic Boundary Value Problems

With 45 Figures and 53 Tables

Springer

Mario Bebendorf
Universität Leipzig
Mathematisches Institut
Fakultät für Mathematik und Informatik
Johannisgasse 26
04103 Leipzig
Germany
bebendorf@math.uni-leipzig.de

ISBN 978-3-540-77146-3 e-ISBN 978-3-540-77147-0

Lecture Notes in Computational Science and Engineering ISSN 1439-7358

Library of Congress Control Number: 2008925088

Mathematics Subject Classification (2000): 65D05, 65D15, 65F05, 65F10, 65F50, 65N22, 65N30, 65N38, 65R20, 65Y05, 65Y20

Cover design: deblik, Berlin

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

spinger.com

To Sophia.

Preface

Hierarchical matrices are a subclass of usual matrices which are data-sparse in the sense that they allow to represent large-scale fully populated matrices with linear or logarithmic-linear complexity. In addition to storing such matrices, approximations of the usual matrix operations can be computed with logarithmic-linear complexity, which can be exploited to setup approximate preconditioners in an efficient and convenient way.

Besides the algorithmic aspects of hierarchical matrices, the main aim of this book is to present their theoretical background in the context of elliptic boundary value problems. The emphasis is laid on robustness with respect to nonsmooth coefficients in the differential operator. Furthermore, the book presents in full detail the adaptive cross approximation method for the efficient treatment of integral operators with non-local kernel functions.

The book originates from a habilitation thesis that was accepted in January 2007 by the Faculty of Mathematics and Computer Science of the University of Leipzig. It is intended to present the vast material on the topic of hierarchical matrices in a self-contained way. Hence, it is written for both researchers and students in mathematics and engineering.

Leipzig, February 2008 *M. Bebendorf*

Acknowledgements

I wish to express my gratitude to Prof. Dr. Dr. h.c. Hackbusch for introducing me to the problem of approximating the inverse of finite element stiffness matrices by hierarchical matrices and for many discussions. Furthermore, I wish to thank Prof. Dr. S. Müller for pointing out the general iteration technique used in the proof of Lemma 4.22. I am also thankful for the guest status at the Max-Planck institute in Leipzig after my change to the University of Leipzig.

All numerical examples in this book were computed using a software library[1] which I started to develop during my time at the Max-Planck institute. Some of the numerical examples stem from a cooperation with ABB Schweiz AG in Baden (Switzerland). I wish to thank Prof. Dr. Andjelić and the power device simulation group for the friendly atmosphere since the start of the cooperation in April 2004. The integration of $\mathscr{H}$-matrices into ABB's software was particularly helpful to improve the abilities and the efficiency of the mentioned software. Furthermore, I wish to thank the German Science Foundation (Deutsche Forschungsgemeinschaft) for the financial support in the priority program SPP 1146 "Modellierung inkrementeller Umformverfahren".

[1] A C++ implementation of the $\mathscr{H}$-matrix structure can be obtained from the following website `http://www.mathematik.uni-leipzig.de/~bebendorf/AHMED.html`

Contents

Acronyms

δ_{ij}	Kronecker symbol ($\delta_{ii} = 1$ and $\delta_{ij} = 0$, $i \neq j$)
$\mathcal{O}(\cdot)$	Landau symbol
$\mathcal{P}(M)$	power set of M
ε, $\varepsilon_{\mathcal{H}}$	often used as accuracy
$\mathbb{N}$	the set of non-negative integers
$\mathbb{R}$, $\mathbb{C}$	the set of real and the set of complex numbers
$\lceil x \rceil$	smallest integer which is larger then x
$\lfloor x \rfloor$	largest integer which is smaller then x
I, J	the sets of row and column indices of the global matrix
$\mathbb{R}^{I \times J}$	the space of matrices in the rows I and columns J
$\mathbb{R}^{I \times k}$	the space of matrices in the rows I and columns $\{1, \ldots, k\}$
$\mathbb{C}_k^{I \times J}$	matrices $M \in \mathbb{C}^{I \times J}$ satisfying $\operatorname{rank} M \leq k$; see page 10
tril, triu	lower and upper triangular part of a matrix; see page 20
$\mathcal{F}(A)$	numerical range of $A \in \mathbb{C}^{n \times n}$
$\|\cdot\|_F$, $\|\cdot\|_2$	Frobenius and spectral norm
$\sigma(A)$	the set of eigenvalues of the matrix A
$\rho(A)$	spectral radius of the matrix A
$\operatorname{cond}_2(A)$	spectral condition number of the matrix A
t, s	usually clusters, i.e., subsets of I and J, respectively
t', t^*	son and father cluster of a cluster t
$\lvert t \rvert$	cardinality of $t \subset \mathbb{N}$
$n_{\min}$	minimal cluster size; see page 29
T_I	cluster tree for the index set I; see Definition 1.16
$\mathcal{L}(T_I)$	the set of leaves of the cluster tree T_I
$\deg t$	the degree of $t \in T_I$
$\operatorname{level} t$	the level of $t \in T_I$; see page 29
$L(T_I)$	the depth of the cluster tree T_I
$T_{I \times J}$	block cluster tree for the index set $I \times J$; see page 43
P	a partition of the matrix indices
$\mathcal{F}(t)$	far-field of a cluster t
c_{sp}	sparsity constant; see page 44

c_{id}	idempotency constant; see page 77
c_{sh}	sharing constant; see page 56
η	cluster parameter
b	an index block; usually $b = t \times s$
$G(A)$	matrix graph of A
d_{ij}	distance of i and j within the matrix graph; see page 24
$\operatorname{diam} t$	diameter of a cluster t; see page 24
$\operatorname{dist}(t,s)$	distance of two clusters with the matrix graph; see page 24
Ω, Γ	computational domain and its boundary $\Gamma = \partial\Omega$
$\nu(x)$	unit normal in $x \in \Gamma$
Ω^c	the exterior $\Omega^c := \mathbb{R}^d \setminus \overline{\Omega}$ of Ω
d	the spatial dimension
∂_i	the partial derivative with respect to the ith variable
$\mathfrak{J}(u)$	Jacobian of u
$\lvert\alpha\rvert$	$\lvert\alpha\rvert = \alpha_1 + \cdots + \alpha_d$ for a multi-index $\alpha \in \mathbb{N}_0^d$
$\operatorname{diam} X$	the diameter of the set $X \subset \mathbb{R}^d$; see page 22
ρ_X, ξ_X	Chebyshev radius and center of the set X; cf. page 120
$\operatorname{dist}(X,Y)$	the distance of two sets $X, Y \subset \mathbb{R}^d$; see page 22
$\operatorname{conv} X$	the convex hull of $X \subset \mathbb{R}^d$
$\operatorname{int} X$	the interior $\operatorname{int} X := \mathbb{R}^d \setminus (\overline{\mathbb{R}^d \setminus X})$ of $X \subset \mathbb{R}^d$
$\dot{\bigcup}$	the disjoint union
$\mu(M)$	measure of the m-dimensional manifold $M \subset \mathbb{R}^d$
$B_r(x)$	Euclidean ball with radius r and center x in $\mathbb{R}^d$
ω_d	the volume of the unit ball in $\mathbb{R}^d$
ω'_d	the surface of the unit sphere in $\mathbb{R}^d$
$\operatorname{supp} u$	the support $\operatorname{supp} u = \overline{\{x \in \mathbb{R}^d : u(x) \neq 0\}}$ of u
Π_k	space of algebraic polynomials of degree at most k
$\mathfrak{F}$	Fourier transform
$L^2(\Omega)$	Lebesgue space of twice integrable functions
$H^k(\Omega)$	Sobolev space of order k
$\mathfrak{I}_p$	interpolation operator of pth order
n	number of degrees of freedom
h	discretization parameter; see page 194
$\mathscr{T}_h$	finite element grid
V_h	finite element space
φ_i, ψ_j	finite element basis and test functions
$\Lambda_{1,i}$, $\Lambda_{2,j}$	localizers; see page 108
$\mathscr{J}$	natural injection $\mathscr{J} : \mathbb{R}^n \to V_h$

Introduction

The simulation of physical phenomena arising in applications from science and engineering often leads to integral equations or to boundary value problems of elliptic partial differential equations. Since such problems usually cannot be solved explicitly, their numerical solution is done by approximating the exact solution from finite-dimensional spaces and solving the resulting problems on a digital computer. The success of numerical simulations together with the rapid development of computers has led to a constant demand for higher accuracy and more sophisticated tasks. In order to be able to satisfy these requirements from industrial applications, the number of degrees of freedom n of the finite-dimensional approximations has to be large enough. The typical size of n in today's computations can reach several millions. This size will further increase in the future since many interesting and important problems still cannot be treated by existing methods.

Numerical schemes which require a number of operations that scales asymptotically like n^2 or higher will not be acceptable in the future. If the next generation of computers has twice the memory, users want to solve problems that are twice as large. However, an algorithm of quadratic complexity requires four times the number of operations, which actually doubles the execution time if we presume that the CPU speed of future processors increases in the same way as the memory does. It is therefore of particular importance to develop numerical schemes which can solve the problem with almost linear complexity, i.e., with a complexity which is linear up to logarithmic factors.

The finite-dimensional approximation is usually done by *finite element* or *finite difference* methods. The latter lead to linear systems of algebraic equations whose solution is usually the bottleneck of the whole simulation. The coefficient matrices arising from the discretization of differential operators are sparse; i.e., only a bounded number of entries per row or column do not vanish. Such linear systems are preferably solved by *iterative methods*, which in contrast to *direct methods* do not change the sparsity of the matrix during the computation. The usage of modern *Krylov subspace methods* requires only the ability to multiply the coefficient matrix by a given vector. The convergence of the latter methods, however, is determined by the spectrum of the matrix, which for discretizations of the considered problems

behaves in such a way that the convergence rate of iterative methods deteriorates for large n. Hence, so-called *preconditioning*, i.e., multiplying the linear system by a non-singular matrix which improves the spectral properties of the coefficient matrix, is unavoidable. Although finding appropriate preconditioners seems to be a purely algebraic problem at first glance, efficient preconditioners usually rely on the analytic background of the linear system.

One of the most efficient and best known solution strategies are *multigrid methods* [125, 46], which use an approximation of the solution on a coarse grid to compute a better approximation on the next finer grid. These methods are able to solve elliptic boundary value problems with asymptotic complexity of optimal order n. They can either be used as an iterative solver, or they can be used for preconditioning other iterations. From the point of efficiency, multigrid methods hence seem to satisfy the requirements of future applications. However, these methods are not robust; i.e., their efficiency depends on many factors such as the geometry, its discretization, the coefficients of the differential operator and so on. Although considerable advancements have been achieved in adapting geometric multigrid methods to any kind of problem, for the application of numerical methods in practical computations it is often expected that the method works in a black-box manner, i.e., with minimal user interaction. In order to account for this requirement, so-called *algebraic multigrid methods* [220] have been introduced which try to achieve the robustness by mostly heuristic strategies.

Multigrid methods can be regarded as the application of an approximate inverse. Robustness, however, can be expected in general only if the exact inverse of the discrete operator is used for solving the problem. The inverse is a fully populated matrix which requires $\mathcal{O}(n^3)$ operations for its computation. A very popular idea to avoid fully populated matrices are approximations of the inverse, so-called *sparse approximate inverses* (SPAI), which possess the same sparsity pattern as the discrete differential operator. The *incomplete LU decomposition* (ILU) is based on similar ideas; see [223]. Although the construction of these preconditioners is attractively fast, their preconditioning effect is limited due to the unnatural restriction of fully populated matrices to sparse matrices.

With the advent of data-sparse matrix representations, it has become possible to treat fully populated matrices with almost linear complexity. The introduction of *hierarchical matrices* ($\mathcal{H}$-matrices[2]) by Hackbusch et al. [127, 132] has paved the way to methods which have almost linear complexity and which are robust. Hierarchical matrices are an algebraic structure which represent sub-blocks and thus each entry of a fully populated matrix by low-rank matrices. In addition to efficiently storing matrices, $\mathcal{H}$-matrices provide approximate variants of the usual matrix operations such as addition, multiplication, and inversion with almost linear complexity. Many of the existing fast methods are based on multi-level structures; see, for instance, *sparse grids* [262, 54] and *hierarchical bases* [261]. In contrast to multigrid methods, the efficiency of $\mathcal{H}$-matrices is based on a suitable hierarchy of partitions of the matrix indices.

[2] $\mathcal{H}$-matrices should not be confounded with H-matrices known from linear algebra; see [149].

While the discretization of differential operators leads to sparse coefficient matrices, discrete integral operators are usually fully populated. It is obvious that already computing the entries of the coefficient matrix does not meet the above requirements. Hence, various efficient numerical schemes for the efficient treatment of nonlocal operators have become widely popular in the last decades. The origin of one class of methods are algorithms [6, 14] for the efficient evaluation of many-body interactions. Well known are the *fast multipole method* [215] and the *panel clustering method* [137, 138], to name just a view. $\mathscr{H}$-matrices and *mosaic-skeletons* [252] can be regarded as algebraic generalizations of such methods. A second class are *wavelet compression techniques* [3], which lead to sparse and asymptotically well-conditioned approximations of the coefficient matrix. All previous methods approximate the discrete integral operator in their specific way. This additional error has an insignificant contribution as long as it is of the order of the discretization error. A direct solution with standard methods is prohibitively expensive. The iterative solution of linear systems arising from the discretization of integral operators is usually less sensitive with respect to the number of degrees of freedom. It will actually be seen in the course of this book that the influence of boundary conditions, for instance, on the convergence properties may be more severe than the dependence on n. For such problems it is necessary to have preconditioners which, in addition to the asymptotic dependence on n, are also able to account for large constants.

Aims

The aim of this book is to introduce a new class of approximate preconditioners for discretizations of both differential and integral operators. The basic structure which is used to compute the approximations are hierarchical matrices. Since these matrices can handle fully populated matrices with almost linear complexity, we are enabled to approximate each entry of the inverse of discretizations of differential operators, for instance, with arbitrary accuracy. The main advantage of approximate preconditioners over existing preconditioners is that problem-independent convergence rates can be achieved. Established preconditioning techniques guarantee at most the boundedness of the condition number with respect to n such that the discretization or the coefficients of the operator may still lead to a large number of iterations. High-precision approximations of the solution operator can also be used as direct solvers. The usage as preconditioners of iterative schemes is however more efficient unless systems with many right-hand sides are to be solved.

Any matrix can be approximated by a hierarchical matrix with arbitrary accuracy. Most of the matrices, however, will lead to a complexity of the approximant that is at least as high as the complexity of the original matrix, because the required blockwise rank may grow linearly with n and with the approximation accuracy. One of the main aims of this book is to show that $\mathscr{H}$-matrices can be used to approximate solution operators of elliptic boundary value problems with almost linear complexity. In order to be able to guarantee this complexity of the $\mathscr{H}$-matrix approximants, it is indispensable to analyze the approximation of discrete inverses. It will be proved that

the efficiency of the approximation by $\mathscr{H}$-matrices depends logarithmic-linearly on the number of degrees of freedom n and logarithmically on the accuracy of the approximation. The complexity is influenced neither by the shape of the domain or its discretization nor by the smoothness of the coefficients and at most logarithmically by their sizes. Blockwise low-rank approximants seem to be capable of adapting themselves to the respective problem. Especially the analysis of the dependence of the approximation on the coefficients of the operator is a main concern of this book. It will be seen that L^∞ coefficients are enough for the existence of approximants – a quite surprising result, because the existence of such kind of approximants used to be attributed to the local smoothness of the Green function, which is only locally Lipschitz continuous in the case of L^∞ coefficients.

The approximation theory for both scalar and systems of elliptic partial differential operators will be derived under quite general assumptions. All results can be ascribed to one single principle, the *interior regularity*, which is characteristic for elliptic boundary value problems. This allows to treat also higher order partial differential operators such as the *biharmonic operator*. In addition, it will show that all problems from the class of elliptic operators can be treated by the same kind of hierarchical subdivision of the computational domain. Hence, low-precision $\mathscr{H}$-matrix approximations of the inverse can be expected to provide robust preconditioners which can be computed with almost linear complexity in a purely algebraic manner. The robustness of $\mathscr{H}$-matrices is however paid by the disadvantage that these preconditioners are often slower for standard problems than a preconditioner which is tailored to the specific problem.

A lack in both, practice and theory, is that the approximation by $\mathscr{H}$-matrices requires the grid to be quasi-uniform although special types of non-uniform grids have been shown to be treatable. The reason for this is that general grids cannot be subdivided such that the number of degrees of freedom and their geometric size are halved at the same time. In order to be able to treat especially graded meshes, which arise, for instance, from adaptive refinement, we introduce a matrix partitioning that is based on the matrix graph. A first algebraic approach to the approximation of inverses of general sparse matrices by $\mathscr{H}$-matrices is presented. This approach helps to explain the phenomenon of so-called *weak admissibility*; i.e., the observation that blocks can be approximated by low-rank matrices although they are not admissible in the usual sense, i.e., far enough away from the singularity.

In addition to the existence of $\mathscr{H}$-matrix approximants to the inverse, it will be proved that the factors of the LU decomposition can be approximated by $\mathscr{H}$-matrices. For this purpose we present a proof which is based on an approximation result for the Schur complement. The latter result is particularly important if *domain decomposition methods* are to be accelerated by $\mathscr{H}$-matrices. Although it is not required to use domain decomposition methods in the presence of variable coefficients, the combination of $\mathscr{H}$-matrices with the latter methods provides an efficient means to improve the computation time by parallelization. The existence of $\mathscr{H}$-matrix approximants to the factors L and U of the LU decomposition is of great practical importance, because such approximations can be computed with significantly less numerical effort than the inverse requires. The ideas of nested dissection

LU decomposition will be used to further improve the efficiency of the hierarchical *LU* factorization.

The efficient treatment of integral equations by $\mathscr{H}$-matrices is a second major concern of this book. Integral equations may arise directly from physical modelling, but they may also result from reformulating boundary value problems under suitable conditions as boundary integral equations. This reformulation is of particular importance if exterior boundary value problems are to be solved. In [17, 18, 32] we have presented a method, the *adaptive cross approximation* (ACA), which can be used to construct $\mathscr{H}$-matrix approximants from few of the original matrix entries. One of the advantages of ACA over the established multipole methods is that existing computer codes can be used to compute the original matrix entries, while multipole methods require a complete recoding. In this book we present a short proof which treats Nyström, collocation, and Galerkin discretizations in a unified way. From the convergence analysis it will be seen that ACA is quasi-optimal in the sense that the constructed low-rank matrices are optimal up to constants. Just as in the case of discretizations of differential operators, the complete theory will be derived from interior regularity. Applications of ACA to nonsmooth boundaries have shown that a thorough choice of rows and columns is important to satisfy the assumptions of the convergence proof of ACA. We present a pivoting strategy which guarantees that ACA converges reliably. Since the construction of the coefficient matrix is still the most expensive part of the computation, scheduling algorithms for the parallelization of the matrix construction using ACA will be presented. Furthermore, we will present a recompression technique based on ACA which achieves the same order of complexity as fast multipole methods and $\mathscr{H}^2$-matrices.

After constructing the coefficient matrix, preconditioners can be obtained from low-precision *LU* factorizations of the approximants in a purely algebraic way. We will construct preconditioners for mixed boundary value problems with almost vanishing Dirichlet part. Since the coefficient matrices of such problems are close to singular, it is likely that perturbations introduced by the $\mathscr{H}$-matrix approximation change important properties such as the positivity of the matrix. In order to preserve these, we introduce stabilization techniques for the approximation by $\mathscr{H}$-matrices.

Overview

This book can be subdivided into three main parts. In the first part we review the structure of hierarchical matrices. The basic principles which the efficiency of $\mathscr{H}$-matrices is based on are presented in Chap. 1. The first ingredient are matrices of low rank. We review basic properties of low-rank matrices in Sect. 1.1. The second principle is a suitable partition of the matrix into sub-blocks. It will be seen in Sect. 1.3 that a hierarchical matrix partitioning is required for an almost linear complexity. A very basic example of $\mathscr{H}$-matrices, which demonstrates all important effects, can be found in Sect. 1.3.1. The construction of general partitions will be based on *cluster trees*; see Sect. 1.5. It will be proved that a matrix partition which is based

on the matrix graph has the same key properties as usual partitions, although it does not use any geometric information.

In Chap. 2 we estimate the complexity of $\mathscr{H}$-matrices and the complexity of algorithms which can be used to compute approximate matrix operations such as addition, multiplication, and inversion. The efficient multiplication of a matrix by a vector is a central problem when solving linear systems by a Krylov subspace method. In order to improve the execution time, we show how $\mathscr{H}$-matrices can be multiplied by a vector in parallel; see Sect. 2.3. Since adding $\mathscr{H}$-matrices leads to approximation errors, it may happen that important matrix properties such as the positivity are lost. A stabilization technique is presented in Sect. 2.5.1 which guarantees that the sum of two positive definite matrices remains positive definite.

The quality of the underlying matrix partition is crucial for the efficiency of $\mathscr{H}$-matrices. Since the initial matrix partition is generated from a condition on the respective block that is only sufficient, one can expect that it contains redundancies which are to be removed. We analyze the complexity of an algebraic technique which can be used to improve partitions by agglomerating neighboring blocks; see Sect. 2.6. In Sect. 2.9 and Sect. 2.10 we present the hierarchical *LU* decomposition and the hierarchical *QR* decomposition. In Sect. 2.12 we will find sufficient conditions on the precision of $\mathscr{H}$-matrix approximants to guarantee the desired preconditioning effect. It will be seen that arbitrarily bad condition numbers of the original matrix can be equalized by sufficiently accurate approximations. Since the approximation accuracy enters the complexity estimates only logarithmically, the complexity of the preconditioner will depend only logarithmically on the condition number of the discrete elliptic operator.

In this first part we regard $\mathscr{H}$-matrices from a purely algebraic point of view. Hence, all complexity estimates will assume that the blockwise rank is bounded. In Chap. 3 and Chap. 4, $\mathscr{H}$-matrices will be applied to boundary element and finite element discretizations of elliptic operators. By restricting the class of problems it will be possible to prove bounds on the required rank of the $\mathscr{H}$-matrix approximation thereby proving almost linear complexity of the computed $\mathscr{H}$-matrices. After giving evidence that elliptic problems lead to asymptotically smooth singularity functions in Sect. 3.2, we show in Sect. 3.3 that asymptotically smooth kernel functions can locally be approximated by degenerate kernels. This, in turn, can be used to define low-rank matrix approximants of discrete integral operators. In Sect. 3.4 we prove the convergence of the adaptive cross approximation method. The storage complexity of ACA generated $\mathscr{H}$-matrices can be brought down to level of $\mathscr{H}^2$-matrices using the recompression technique presented in Sect. 3.5. In order to demonstrate the reliability of ACA, we apply it to complicated geometries in Sect. 3.6. We report the results of numerical experiments with the hierarchical *LU* preconditioner and demonstrate the efficiency and ease of use of these methods when they are applied to real applications from electromagnetism and from linear elasticity.

In Chap. 4 we first prove the fundamental result that the inverse of discrete differential operators can be approximated by $\mathscr{H}$-matrices with logarithmic-linear complexity. Based on the existence results for the inverse we show in Sect. 4.2

that $\mathscr{H}$-matrices can be used to efficiently treat Schur complements. This result paves the way to complexity estimates for the approximation of the factors L and U of the LU decomposition. The ideas of nested dissection can be adopted in order to improve and parallelize the hierarchical matrix LU factorization algorithm; see Sect. 4.5. As we have mentioned, $\mathscr{H}$-matrix approximants are mainly used for the purpose of preconditioning in this book. Their efficiency is demonstrated in Sect. 4.4 when low-precision hierarchical LU preconditioners are applied to finite element discretizations. We compare the LU preconditioner with current sparse direct solvers. The property that the same algorithm can be used for all operators from the class of elliptic boundary value problems is helpful when nonlinear problems are to be treated efficiently. Such problems are usually solved by Newton's method, which approximates the nonlinearity by a sequence of linear problems with possibly varying properties of the operators. In Sect. 4.6 we present an $\mathscr{H}$-matrix accelerated version of Broyden's method which is based on explicitly updating the LU decomposition if a rank-1 matrix is added.

Chapter 1
Low-Rank Matrices and Matrix Partitioning

In this chapter we introduce the two principles the efficiency of hierarchical matrices is based on. In Sect. 1.1 we consider low-rank matrices and their approximation properties. Compared with general matrices the members of this subclass contain less independent information, which can be exploited to reduce their storage requirement and to speed up arithmetic operations such as addition, multiplication, and even their singular value decomposition. Matrices usually cannot be represented globally by low-rank matrices. A generalization are semi-separable matrices (cf. Sect. 1.2) which are low-rank on each block in the upper or lower triangular part. Semi-separable matrices can be used to treat boundary value problems of one spatial dimension. For boundary value problems in general spatial dimension the matrix has to be appropriately partitioned using the techniques from Sects. 1.3–1.5.

1.1 Low-Rank Matrices

Let $m,n \in \mathbb{N}$ and $A \in \mathbb{C}^{m\times n}$ be a matrix. The range of A is defined as the result of multiplying A by any vector from $\mathbb{C}^n$

$$\mathrm{Im}A := \{Ax \in \mathbb{C}^m, x \in \mathbb{C}^n\}.$$

The rank of A is the dimension of its range

$$\mathrm{rank}A := \dim \mathrm{Im}A.$$

We will make use of the following well-known relations; see for instance [148].

Theorem 1.1. *Let $m,n,k \in \mathbb{N}$. Then it holds that*

(i) $\mathrm{rank}A \leq \min\{m,n\}$ *for all* $A \in \mathbb{C}^{m\times n}$*;*
(ii) $\mathrm{rank}(AB) \leq \min\{\mathrm{rank}A, \mathrm{rank}B\}$ *for all* $A \in \mathbb{C}^{m\times p}$ *and all* $B \in \mathbb{C}^{p\times n}$*;*
(iii) $\mathrm{rank}(A+B) \leq \mathrm{rank}A + \mathrm{rank}B$ *for all* $A,B \in \mathbb{C}^{m\times n}$.

M. Bebendorf, *Hierarchical Matrices*. Lecture Notes in Computational Science and Engineering 63,

We denote the set of matrices $A \in \mathbb{C}^{m\times n}$ having at most k linearly independent rows or columns by

$$\mathbb{C}_k^{m\times n} := \left\{A \in \mathbb{C}^{m\times n} : \operatorname{rank} A \le k\right\}.$$

Note that $\mathbb{C}_k^{m\times n}$ is not a linear space. The rank of the sum of two rank-k matrices is in general only bounded by $2k$; see Theorem 1.1. An important property is that the rank of each sub-block of $A \in \mathbb{C}_k^{m\times n}$ is bounded by k.

Matrices having a rank that is relatively small compared with their dimensions m and n are one of the basic structures the efficiency of hierarchical matrices is based on. Therefore, in the next section we remind the reader of some of their important properties.

1.1.1 Efficient Representation

In this book only two kinds of matrix representations will be considered. One is the (usual) entrywise representation and the other is the outer-product form (1.1). There are many other ways to represent matrices if additional information about their structure is given. For instance, it is sufficient to store only the first row and the first column of a Toeplitz matrix. Such additional information, however, is available only in special situations. Our aim is to exploit properties that are present for a whole class of problems.

Since among the n columns of $A \in \mathbb{C}_k^{m\times n}$ only k are sufficient to represent the whole matrix by linear combination, the entrywise representation of A contains redundancies which can be removed by changing to another kind of representation.

Theorem 1.2. *A matrix $A \in \mathbb{C}^{m\times n}$ belongs to $\mathbb{C}_k^{m\times n}$ if and only if there are matrices $U \in \mathbb{C}^{m\times k}$ and $V \in \mathbb{C}^{n\times k}$ such that*

$$A = UV^H. \tag{1.1}$$

The representation (1.1) of matrices from $\mathbb{C}_k^{m\times n}$ is called **outer-product form**. If u_i, v_i, $i = 1,\dots,k$, denote the columns of U and V, respectively, then (1.1) can be equivalently written as

$$A = \sum_{i=1}^{k} u_i v_i^H.$$

Hence, instead of storing the $m \cdot n$ entries of $A \in \mathbb{C}_k^{m\times n}$, we can equally store the vectors u_i, v_i, $i = 1,\dots,k$, which require $k(m+n)$ units of storage.

In addition to reducing the storage requirements, the outer-product form (1.1) also facilitates matrix-vector multiplications

$$Ax = UV^H x = U(V^H x),$$

i.e, instead of computing the update $y := y + Ax$ in the entrywise way, A can alternatively be multiplied by x using the following two-step procedure:

1. define $z := V^H x \in \mathbb{C}^k$;
2. compute $y := y + Uz$.

Hence, instead of $2m \cdot n$ arithmetic operations which are required in the entrywise representation, the outer-product form amounts to $2k(m+n) - k$ operations. Note that these fundamental properties are also exploited when dealing, for instance, with Householder reflections.

Assume for a moment that $m = n$. Although the outer-product form replaces one dimension in the complexity of matrices from $\mathbb{C}_k^{m\times n}$ by $2k$, i.e., instead of $m \cdot n = m^2$ we have $k(m+n) = 2km$, this representation might not be advantageous compared with the entrywise representation. If we are dealing, for instance, with full-rank matrices, i.e., $k = m$, $k(m+n)$ will have the size $2m^2$, which is twice as large as the expression $m \cdot n$ appearing as the complexity of the entrywise representation. By the following definition we characterize matrices for which the outer-product form is advantageous compared with the entrywise representation.

Definition 1.3. A matrix $A \in \mathbb{C}_k^{m\times n}$ is called a **matrix of low rank** if

$$k(m+n) < m \cdot n. \tag{1.2}$$

Thus, low-rank matrices will always be represented in outer-product form (1.1), while the entrywise representation will be used for all other matrices.

If $A, B \in \mathbb{C}^{n\times n}$ are non-singular matrices and B arises from A by adding a matrix $UV^H \in \mathbb{C}_k^{n\times n}$, then provided that $I + V^H A^{-1} U$ is non-singular also the inverse of B arises from the inverse of A by adding a matrix from $\mathbb{C}_k^{n\times n}$:

$$B^{-1} = A^{-1} - A^{-1}U(I + V^H A^{-1} U)^{-1} V^H A^{-1}. \tag{1.3}$$

The previous identity is usually referred to as the **Sherman-Morrison-Woodbury formula**; cf. [148].

Besides arithmetic operations also the norm of a matrix is often required. The **Frobenius norm**

$$\|A\|_F := \sqrt{\operatorname{trace} A^H A} = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2}$$

of $A \in \mathbb{C}_k^{m\times n}$ can be computed with $2k^2(m+n)$ operations due to

$$\|UV^H\|_F^2 = \sum_{i,j=1}^{k} (u_i^H u_j)(v_i^H v_j). \tag{1.4}$$

Similarly, the **spectral norm**

$$\|A\|_2 := \sqrt{\rho(A^H A)} = \sqrt{\rho(VU^H UV^H)} = \sqrt{\rho(U^H UV^H V)},$$

where $\rho(A)$ denotes the spectral radius of A, can be evaluated with $\mathcal{O}(k^2(m+n))$ arithmetic operations by first computing the $k \times k$ matrices $U^H U$ and $V^H V$ and then computing the largest eigenvalue of the product. Note that this is advantageous compared with the entrywise way of computing the Frobenius or the spectral norm only if $k^2(m+n) < m \cdot n$, which is a stronger condition on the size of k than condition (1.2).

Sometimes it is useful that U and V have orthonormal columns; i.e., it holds that $U^H U = I_k = V^H V$. In this case we have to introduce an additional coefficient matrix $X \in \mathbb{R}^{k \times k}$ and replace (1.1) by the **orthonormal outer-product form**

$$A = UXV^H. \tag{1.5}$$

If the previous representation is employed, the computation of the Frobenius norm simplifies to

$$\|UXV^H\|_F = \|X\|_F = \sqrt{\sum_{i,j=1}^{k} |x_{ij}|^2},$$

which requires only $\mathcal{O}(k^2)$ operations. Since the spectral norm is unitarily invariant, too, we have

$$\|UXV^H\|_2 = \|X\|_2,$$

which leads to the computation of the largest eigenvalue of a $k \times k$ matrix.

1.1.2 Adding and Multiplying Low-Rank Matrices

In addition to multiplying low-rank matrices by vectors, also the problem of efficiently multiplying and adding low-rank matrices will occur.

We first consider the multiplication of two low-rank matrices $A \in \mathbb{C}_{k_A}^{m \times p}$ and $B \in \mathbb{C}_{k_B}^{p \times n}$ in outer-product representation $A = U_A V_A^H$ and $B = U_B V_B^H$. As we know from Theorem 1.1, the rank of the product AB is bounded by $\min\{k_A, k_B\}$. Hence, the outer-product form will be advantageous for the product AB as well. There are two possibilities for computing $AB = UV^H$:

(a) $U := U_A(V_A^H U_B)$ and $V := V_B$ using $2k_A k_B(m+p) - k_B(m+k_A)$ operations;
(b) $U := U_A$ and $V := V_B(U_B^H V_A)$ using $2k_A k_B(p+n) - k_A(n+k_B)$ operations.

Depending on the quantities k_A, k_B, m, and n, either representation should be chosen.

If exactly one of the matrices A or B is stored entrywise, say B, we have the following outer-product representation of AB:

$$AB = UV^H,$$

where $U := U_A \in \mathbb{C}^{m \times k_A}$ and $V := B^H V_A \in \mathbb{C}^{n \times k_A}$. This requires $k_A(2p-1)n$ operations.

Remark 1.4. The previous complexity estimate seems to indicate that the number of operations is quadratic with respect to the dimension. However, the situation that B is not low-rank will occur only if its dimensions are bounded by a constant.

If the orthonormal outer-product form (1.5) is used, i.e., $A = U_A X_A V_A^H \in \mathbb{C}_k^{m\times p}$ and $B = U_B X_B V_B^H \in \mathbb{C}_k^{p\times n}$, then

$$AB = U_A X V_B^H, \quad \text{where } X := X_A V_A^H U_B X_B.$$

Notice that $X \in \mathbb{C}^{k\times k}$ can be computed with $k^2(2p+2k-3)$ operations. In the case that $A \in \mathbb{C}^{m\times p}$ is stored entrywise, the orthonormal outer-product form can be reestablished for $AU_B X_B V_B^H$ by a QR decomposition of AU_B.

If two matrices $A \in \mathbb{C}_{k_A}^{m\times n}$ and $B \in \mathbb{C}_{k_B}^{m\times n}$ having the representations $A = U_A V_A^H$ and $B = U_B V_B^H$ are to be added, the sum $A+B$ will have the following outer-product representation

$$A + B = UV^H$$

with $U := [U_A, U_B] \in \mathbb{C}^{m\times k}$ and $V := [V_A, V_B] \in \mathbb{C}^{n\times k}$, which guarantees that

$$\operatorname{rank}(A+B) \le k_A + k_B =: k. \tag{1.6}$$

Hence, apart from the reorganization of the data structure, no numerical operations are required for adding two matrices in outer-product form.

A lower bound than (1.6) for the rank of $A+B$ cannot be found for general low-rank matrices. Hence, the rank of $A+B$ will be considerably larger than the ranks of A and B although $A+B$ might be close to a matrix of a much smaller rank.

1.1.3 Approximation by Low-Rank Matrices

Although matrices usually have full rank, they can often be approximated by matrices having a much lower rank. The following theorem (see [231, 85, 188]) states that the closest matrix in $\mathbb{C}_k^{m\times n}$ to a given matrix from $\mathbb{C}^{m\times n}$, $m \ge n$, can be obtained from the singular value decomposition (SVD) $A = U\Sigma V^H$ with $U^H U = I_n = V^H V$ and a diagonal matrix $\Sigma \in \mathbb{R}^{n\times n}$ with entries $\sigma_1 \ge \ldots \ge \sigma_n \ge 0$. Interestingly, this result is valid for any unitarily invariant norm.

Theorem 1.5. *Let the SVD $A = U\Sigma V^H$ of $A \in \mathbb{C}^{m\times n}$, $m \ge n$, be given. Then for $k \in \mathbb{N}$ satisfying $k \le n$ it holds that*

$$\min_{M\in\mathbb{C}_k^{m\times n}} \|A - M\| = \|A - A_k\| = \|\Sigma - \Sigma_k\|, \tag{1.7}$$

where $A_k := U\Sigma_k V^H \in \mathbb{C}_k^{m\times n}$ and $\Sigma_k := \operatorname{diag}(\sigma_1, \ldots, \sigma_k, 0, \ldots, 0) \in \mathbb{R}^{n\times n}$.

Note that the approximant $U\Sigma_k V^H$ has the representation (1.5). If the outer-product representation is preferred, either U or V has to be multiplied by Σ_k.

If the spectral norm $\|\cdot\|_2$ is used in the previous theorem, then

$$\|A - A_k\|_2 = \sigma_{k+1},$$

while in the case of the Frobenius norm one has

$$\|A - A_k\|_F^2 = \sum_{\ell=k+1}^{n} \sigma_\ell^2.$$

Hence, the approximation error in unitarily invariant norms can be obtained by inspecting the singular values of A.

The information about the error $\|A - A_k\| = \|\Sigma - \Sigma_k\|$ can be used in two different ways. If the maximum rank k of the approximant is prescribed, one gets to know the associated error. If on the other hand a relative accuracy $\varepsilon > 0$ of the approximant A_k is prescribed (say with respect to the spectral norm), i.e.,

$$\|A - A_k\|_2 < \varepsilon \|A\|_2,$$

then due to (1.7) the required rank $k(\varepsilon)$ is given by

$$k(\varepsilon) := \min\{k \in \mathbb{N} : \sigma_{k+1} < \varepsilon \sigma_1\}.$$

Remark 1.6. Later on it will be seen that appropriate blocks of matrices arising from the discretization of elliptic operators will have the property that they can be approximated by exponentially converging low-rank matrices S_k; i.e.,

$$\|A - S_k\|_2 < q^k \|A\|_2$$

for some $0 < q < 1$. As a consequence of this exponential convergence the singular values

$$\sigma_{k+1} = \|A - A_k\|_2 \leq \|A - S_k\|_2 < q^k \|A\|_2 = q^k \sigma_1$$

of such blocks decay exponentially. Hence, the rank $k(\varepsilon)$ of the approximant will depend only logarithmically on the given accuracy ε. Since the approximant S_k will not be generated from the SVD, its rank cannot be expected to be optimal. In this case S_k can be recompressed, i.e., an approximant with lower rank can be found which also guarantees the prescribed accuracy. A similar situation occurs if the accuracy of the approximant S_k can be reduced. This happens, for instance, when computing low-precision preconditioners from a given matrix.

Instead of the SVD one could also obtain low-rank approximations by a **rank-revealing QR decomposition** (RRQR). Let $k \in \mathbb{N}$, $A \in \mathbb{C}^{m \times n}$, $m \geq n$, with non-increasingly ordered singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. A rank-revealing QR decomposition is a QR decomposition

$$A\Pi = QR$$

with $Q \in \mathbb{C}^{m\times n}$ having orthonormal columns and a permutation matrix $\Pi \in \mathbb{R}^{n\times n}$ such that $R_{11} \in \mathbb{C}^{k\times k}$ in

$$R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix} \in \mathbb{C}^{n\times n},$$

is upper triangular and $\|R_{22}\| = \mathscr{O}(\sigma_{k+1})$. In this case let $U \in \mathbb{C}^{m\times k}$ be the first k columns of Q and $V := \Pi \begin{bmatrix} R_{11} & R_{12} \end{bmatrix}^H$. Then

$$\|A - UV^H\|_2 = \|R_{22}\|_2 \leq c\sigma_{k+1}.$$

The computation using $\mathscr{O}(kmn)$ operations can be done using, for instance, the **Businger-Golub algorithm** [56]. Rank-revealing factorizations, however, are less reliable than the SVD. In order to find the optimum approximant A_k, it remains to compute the singular value decomposition, which requires $\mathscr{O}(mn^2)$ operations for general matrices $A \in \mathbb{C}^{m\times n}$, $m \geq n$. If the given matrix A has low rank and if it is stored in outer-product form, then its SVD can be computed with significantly less operations. How this is done is explained in the following section.

1.1.4 Singular Value Decomposition of Low-Rank Matrices

The singular value decomposition of matrices $\mathbb{C}^{m\times n}$, $m \geq n$, is an expensive operation. From [106, §5.4.5] it can be seen that the cost of computing an SVD for general matrices from $\mathbb{C}^{m\times n}$ is $14mn^2 + 8n^3$ (complex operations). However, for matrices $A = UV^H \in \mathbb{C}_k^{m\times n}$ it is possible to compute an SVD with complexity $\mathscr{O}(k^2(m+n))$. A method based on **Gram matrices** was proposed in [127]. The following method, which was introduced in [18], uses the QR decomposition and is more efficient.

Assume we have computed the QR decompositions

$$U = Q_U R_U \quad \text{and} \quad V = Q_V R_V$$

of $U \in \mathbb{C}^{m\times k}$ and $V \in \mathbb{C}^{n\times k}$, respectively. According to [106, §5.2.9], this can be done with $4k^2(m+n) - \frac{8}{3}k^3$ operations. The outer product $R_U R_V^H$ of the two $k \times k$ upper triangular matrices R_U and R_V is then decomposed using the SVD

$$R_U R_V^H = \hat{U}\hat{\Sigma}\hat{V}^H.$$

Computing $R_U R_V^H$ needs $\frac{k}{3}(2k^2 + \frac{11}{2}k - 1)$ operations, and the cost of the SVD amount to $22k^3$ operations. Since $Q_U\hat{U}$ and $Q_V\hat{V}$ both have orthonormal columns,

$$A = UV^H = (Q_U\hat{U})\hat{\Sigma}(Q_V\hat{V})^H$$

is an SVD of A. Together with the products $Q_U\hat{U}$ and $Q_V\hat{V}$, which require $k(2k-1)$ $(m+n)$ operations, the number of arithmetic operations of the SVD of a rank-k matrix sum up to

QR decomposition of U and V	$4k^2(m+n)-\frac{8}{3}k^3$
Computing $R_U R_V^H$	$\frac{2}{3}k^3+\frac{11}{6}k^2-\frac{1}{3}k$
SVD of $R_U R_V^H$	$22k^3$
Computing $Q_U\hat{U}$ and $Q_V\hat{V}$	$k(2k-1)(m+n)$
	$\sim 6k^2(m+n)+20k^3$

operations.

1.1.5 Approximate Addition of Low-Rank Matrices

When computing the sum of two low-rank matrices, we have to deal with the problem that $\mathbb{C}_k^{m\times n}$ is not a linear space. If two matrices $A\in\mathbb{C}_{k_A}^{m\times n}$ and $B\in\mathbb{C}_{k_B}^{m\times n}$ having the representations $A=U_AV_A^H$ and $B=U_BV_B^H$ are to be added, the sum

$$A+B=UV^H$$

with $U:=[U_A,U_B]\in\mathbb{C}^{m\times k}$ and $V:=[V_A,V_B]\in\mathbb{C}^{n\times k}$ might however be close to a matrix of a much smaller rank. In this case the sum of two low-rank matrices can be truncated to rank k. This truncation will be referred to as the **rounded addition**. Employing the SVD of low-rank matrices from the previous section, the rounded addition can be performed with $\mathcal{O}((k_A+k_B)^2(m+n))$ operations.

Theorem 1.7. *Let $A\in\mathbb{C}_{k_A}^{m\times n}$, $B\in\mathbb{C}_{k_B}^{m\times n}$, and $k\in\mathbb{N}$ with $k\le k_A+k_B$. Then a matrix $S\in\mathbb{C}_k^{m\times n}$ satisfying*

$$\|A+B-S\|=\min_{M\in\mathbb{C}_k^{m\times n}}\|A+B-M\|$$

with respect to any unitarily invariant norm $\|\cdot\|$ can be computed with $6(k_A+k_B)^2(m+n)+20(k_A+k_B)^3$ operations.

In some applications it may also occur that the sum of several low-rank matrices $A_i\in\mathbb{C}_{k_i}^{m\times n}$, $i=1,\dots,\ell$, has to be rounded. In this case, the complexity analysis reveals a factor $(\sum_{i=1}^{\ell}k_i)^2$ in front of $m+n$. In order to avoid this, we gradually compute the rounded sum pairwise. In this case the number of operations reduces to

$$6\sum_{i=1}^{\ell-1}(k_i+k_{i+1})^2(m+n)+20\sum_{i=1}^{\ell-1}(k_i+k_{i+1})^3$$

for the price of losing the best approximation property.

1.1.5.1 Exploiting Orthogonality for the Rounded Addition

The rounded addition is the most time-consuming part in the arithmetic of hierarchical matrices. Therefore, it is worth investigating other algorithms for the rounded addition. In Theorem 1.7 it is assumed that A and B enter the computation through the outer-product representation (1.1). If the representation (1.5) is used instead, the numerical effort can be further reduced. Assume that

$$A = U_A X_A V_A^H \quad \text{and} \quad B = U_B X_B V_B^H,$$

where U_A, V_A, U_B, and V_B have orthonormal columns and $X_A \in \mathbb{R}^{k_A \times k_A}$, $X_B \in \mathbb{R}^{k_B \times k_B}$. Then

$$A + B = [U_A, U_B] \begin{bmatrix} X_A & \\ & X_B \end{bmatrix} [V_A, V_B]^H.$$

Assume that $k_A \geq k_B$. In order to reestablish a representation of type (1.5), we have to orthogonalize the columns of the matrices $[U_A, U_B]$ and $[V_A, V_B]$. Let

$$Z_U := U_A^H U_B \in \mathbb{C}^{k_A \times k_B} \quad \text{and} \quad Y_U := U_B - U_A Z_U \in \mathbb{C}^{m \times k_B}.$$

Furthermore, let $Q_U R_U = Y_U$, $Q_U \in \mathbb{C}^{m \times k_B}$, be a QR decomposition of Y_U. Then

$$[U_A, U_B] = \begin{bmatrix} U_A, Q_U \end{bmatrix} \begin{bmatrix} I & Z_U \\ & R_U \end{bmatrix}$$

is a QR decomposition of $[U_A, U_B]$, because the columns of U_A are already orthonormal. Similarly,

$$[V_A, V_B] = \begin{bmatrix} V_A, Q_V \end{bmatrix} \begin{bmatrix} I & Z_V \\ & R_V \end{bmatrix}$$

is a QR decomposition of $[V_A, V_B]$, where $Z_V := V_A^H V_B \in \mathbb{C}^{k_A \times k_B}$ and $Q_V R_V = Y_V$ is a QR decomposition of $Y_V := V_B - V_A Z_V \in \mathbb{C}^{n \times k_B}$. We obtain

$$A + B = [U_A, Q_U] \begin{bmatrix} X_A + Z_U X_B Z_V^H & Z_U X_B R_V^H \\ R_U X_B Z_V^H & R_U X_B R_V^H \end{bmatrix} [V_A, Q_V]^H.$$

From this point on one proceeds in the same way as for the SVD of low-rank matrices.

The orthogonalization of $[U_A, U_B]$ using the previous method requires

Computing Z_U	$k_A k_B (2m-1)$
Computing Y_U	$2k_A k_B m$
Decomposing Y_U	$4k_B^2 m - \frac{4}{3}k_B^3$
	$4k_B(k_A+k_B)m - k_B(k_A + \frac{4}{3}k_B^2)$

operations, while applying the QR factorization algorithm to $[U_A, U_B]$ needs $4(k_A + k_B)^2 m - \frac{4}{3}(k_A + k_B)^3$ operations. If $k := k_A = k_B$, then the proposed variant requires $8k^2 m - k^2(\frac{4}{3}k + 1)$ operations, while $16k^2 m - 10\frac{2}{3}k^3$ operations are needed to

decompose $[U_A, U_B]$ directly. In Table 1.1 we compare the two ways of computing the rounded addition for five problem sizes. The presented CPU times are the times for 10000 additions with accuracy $\varepsilon = 1_{10}-2$. Hence, using the modified addition algorithm almost half of the time can be saved by exploiting the property that the columns of U_A and V_A are already orthonormal.

Table 1.1 10000 executions of the old and the new rounded addition.

$m \times n$	k_A	k_B	time old	time new	gain
200×100	8	5	5.23s	4.78s	9%
300×200	10	7	12.57s	8.51s	32%
400×200	11	8	16.05s	9.92s	38%
600×300	12	9	30.05s	15.94s	47%
800×400	13	10	45.97s	23.82s	48%

1.1.6 Agglomerating Low-Rank Blocks

We will come across the problem of unifying neighboring blocks to a single one for the purpose of saving memory. This operation will be referred to as **agglomeration**. Assume a 2×2 block matrix

$$\begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \approx UXV^H$$

consisting of four low-rank matrices $A_i = U_i X_i V_i^H$ with U_i, V_i, $i = 1, \ldots, 4$, each having k orthonormal columns is to be approximated by a single matrix $A = UXV^H \in \mathbb{C}_k^{m \times n}$. Since

$$\begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} = \begin{bmatrix} A_1 & \\ & \end{bmatrix} + \begin{bmatrix} & A_2 \\ & \end{bmatrix} + \begin{bmatrix} & \\ A_3 & \end{bmatrix} + \begin{bmatrix} & \\ & A_4 \end{bmatrix},$$

this problem may be regarded as a rounded addition of four low-rank matrices. Therefore, a best approximation in $\mathbb{C}_k^{m \times n}$ can be computed using the SVD of low-rank matrices.

Compared with the rounded addition of general low-rank matrices, the presence of zeros should be taken into account. Since

$$\begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} = \hat{U}\hat{X}\hat{V}^H, \quad \text{where } \hat{U} := \begin{bmatrix} U_1 & U_2 & & \\ & & U_3 & U_4 \end{bmatrix}, \quad \hat{V} := \begin{bmatrix} V_1 & & V_3 & \\ & V_2 & & V_4 \end{bmatrix},$$

and $\hat{X} = \operatorname{diag}(X_i, i = 1, \ldots, 4)$, it satisfies to compute the QR decompositions

$$[U_1, U_2] = Q_1 R_1, \quad [U_3, U_4] = Q_2 R_2, \quad [V_1, V_3] = Q_3 R_3, \quad \text{and} \quad [V_2, V_4] = Q_4 R_4.$$

Let $R_i = [R_i', R_i'']$ be partitioned with $R_i', R_i'' \in \mathbb{C}^{2k\times k}$, then

$$\hat{U}\hat{X}\hat{V}^H = \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix} \begin{bmatrix} R_1'X_1R_3'^H & R_1''X_2R_4'^H \\ R_2'X_3R_3''^H & R_2''X_4R_4''^H \end{bmatrix} \begin{bmatrix} Q_3^H & \\ & Q_4^H \end{bmatrix}.$$

The number of arithmetic operations can be estimated as

Computing the QR decompositions	$16k^2(m+n) - 15\frac{1}{3}k^3$
Computing $R_1'X_1R_3'^H$, $R_1''X_2R_4'^H$, $R_2'X_3R_3''^H$, $R_2''X_4R_4''^H$	$2k^3$, $4k^3$, $4k^3$, $6k^3$
Computing the SVD of $R_{\hat{U}}\hat{X}R_{\hat{V}}^H$	$22(4k)^3$
Building the unitary factors	$8k^2(m+n)$
	$\sim\ 24k^2(m+n) + 1408\frac{2}{3}k^3$

The amount of operations can be reduced if each of the matrices $[A_1, A_2]$ and $[A_3, A_4]$ is agglomerated before agglomerating the results. However, in this case we cannot expect to obtain a best approximation in $\mathbb{C}_k^{m\times n}$.

1.2 Structured Low-Rank Matrices

We have seen in the previous section that low-rank matrices are an efficient means to approximate matrices having exponentially decaying singular values. A global approximation by low-rank matrices, however, is possible in only few applications. A generalization of the class of low-rank matrices are *structured low-rank matrices*.

Definition 1.8. A matrix $A \in \mathbb{C}^{m\times n}$ is called **matrix of structured rank** k if matrices $U \in \mathbb{C}^{m\times k}$, $V \in \mathbb{C}^{n\times k}$ and numbers $j_i \in \{1,\dots,n\}$, $i = 1,\dots,m$, satisfying $j_i \le j_{i+1}$ can be found such that

$$a_{ij} = \begin{cases} v_j^H u_i, & j \le j_i, \\ 0, & \text{else.} \end{cases}$$

Here, $u_i, v_i \in \mathbb{C}^k$ denote the ith rows of U and V, respectively.

If the j_i, $i = 1,\dots,m$, are not monotonously increasing, then the rows have to be reordered such that this assumption is fulfilled.

Structured rank-k matrices require $k(m+n)$ units of storage for the matrices U, V and m units for the indices j_i. Although such matrices usually have full rank, they can be multiplied by a vector x with $\mathcal{O}(k(m+n))$ operations due to the following observation:

$$y_i := (Ax)_i = \sum_{j=1}^{j_i} v_j^H u_i x_j = s_i^H u_i,$$

where $s_i := \sum_{j=1}^{j_i} \overline{x}_j v_j = s_{i-1} + \sum_{j=j_{i-1}+1}^{j_i} \overline{x}_j v_j$.

An obvious alternative representation of structured rank-k matrices is

$$A = \sum_{\ell=1}^{k} \operatorname{diag}(u_\ell)\, E \operatorname{diag}(\overline{v}_\ell),$$

where $u_\ell \in \mathbb{C}^m$ and $v_\ell \in \mathbb{C}^n$ now denote the ℓth column of U and V, respectively, and the entries of E are given by

$$e_{ij} = \begin{cases} 1, & j \le j_i, \\ 0, & \text{else.} \end{cases}$$

Before we define what (p,q)-semi-separable matrices are, we introduce (similar to the respective MATLAB commands) the notation $\operatorname{triu}(A,p)$ for the matrix which results from A by setting the diagonals below the pth supdiagonal to zero. Analogously, $\operatorname{tril}(A,p)$ denotes the matrix with zero diagonals above the pth subdiagonal of A.

Example 1.9. For

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

we have

$$\operatorname{triu}(A,1) = \begin{bmatrix} & 2 & 3 \\ & & 6 \\ & & \end{bmatrix} \quad \text{and} \quad \operatorname{tril}(A,1) = \begin{bmatrix} & & \\ 4 & & \\ 7 & 8 & \end{bmatrix}.$$

Definition 1.10. A matrix $A \in \mathbb{C}^{n\times n}$ is said to be (p,q)**-semi-separable** if matrices $U,V \in \mathbb{C}^{n\times p}$ and $W,Z \in \mathbb{C}^{n\times q}$ can be found such that

$$A = \operatorname{triu}(UV^H, 0) + \operatorname{tril}(WZ^H, 1).$$

The matrix A is called **diagonal-plus-semi-separable** if

$$A = \operatorname{triu}(UV^H, 1) + D + \operatorname{tril}(WZ^H, 1)$$

with some diagonal matrix $D \in \mathbb{C}^{n\times n}$.

Therefore, semi-separable matrices (see [254]) can be represented as the sum of a structured rank-p and a structured rank-q matrix. Hence, they can be multiplied by a vector with complexity $\mathcal{O}((p+q)n)$ using Algorithm 1.1.

$y := Dx$
for $k = 1,\dots,q$ **do**
  $s := \overline{z}_{1k} x_1$
  for $i = 2,\dots,n$ **do**
    $y_i := y_i + w_{ik} s$
    $s := s + \overline{z}_{ik} x_i$
for $k = 1,\dots,p$ **do**
  $s := \overline{v}_{nk} x_n$
  for $i = n-1,\dots,1$ **do**
    $y_{ik} := y_{ik} + u_{ik} s$
    $s := s + \overline{v}_{ik} x_i$

Algorithm 1.1: Diagonal-plus-semi-separable matrix times a vector.

The multiplication with the Hermitian transposed A^H can be done in the same way after interchanging the roles of U, Z and V, W, respectively. Multiplying semi-separable and general matrices $A \in \mathbb{C}^{m\times n}$ can therefore be done with $\mathcal{O}((p+q)mn)$ arithmetic operations.

Semi-separable matrices can be used to explicitly represent inverses of banded matrices and solution operators of elliptic boundary value problems in one spatial dimension; see Sect. 4.1.1. They can be factored by triangular matrices with linear complexity; cf. [86, 87, 179, 60, 59]. A generalization of semi-separable matrices are **hierarchically semi-separable matrices** [62, 61]. The latter class of matrices is based on a hierarchical matrix partition which, however, is too restrictive to treat problems in more than one spatial dimension. The following section on matrix partitioning will describe how to generate partitions which are appropriate for problems of arbitrary dimension.

1.3 Admissible Partitions

Typically, matrices $A \in \mathbb{C}^{m\times n}$ can be approximated by low-rank matrices only on sub-blocks of an appropriately chosen partition of the matrix indices $I \times J$. Here and in the rest of this book, I and J will be used as the set of row and column indices $I := \{1,\dots,m\}$ and $J := \{1,\dots,n\}$.

Definition 1.11. Let $I, J \subset \mathbb{N}$. A subset $P \subset \mathcal{P}(I \times J)$ of the set of subsets of $I \times J$ is called **partition** if

$$I \times J = \bigcup_{b\in P} b$$

and if $b_1 \cap b_2 \neq \varnothing$ implies $b_1 = b_2$ for all $b_1, b_2 \in P$. The elements $b \in P$ will be called **index blocks** or just blocks.

It is common to denote the ith component of a vector $x \in \mathbb{C}^I$ by x_i. We will use the following generalization. If $t \subset I$, then $x_t \in \mathbb{C}^t$ denotes the restriction of x to the indices in t. Note that $\mathbb{C}^t$ is used in contrast to $\mathbb{C}^{|t|}$ in order to emphasize the index structure of $x \in \mathbb{C}^t$. Analogously, A_{ts} or A_b denotes the restriction of a given matrix $A \in \mathbb{C}^{m\times n}$ to the indices in $b := t \times s$, where $t \subset I$ and $s \subset J$.

The aim of this section is to introduce algorithms for generating partitions P of matrices $A \in \mathbb{C}^{I\times J}$ such that the restriction A_b of A to each block $b \in P$ can either be approximated by a matrix of low rank or is small. When constructing partitions, we have to account for two contrary aims. On one hand the partition has to be fine enough such that most of the blocks can be successfully approximated. On the other hand the number of blocks must be as small as possible in order to be able to guarantee efficient arithmetic operations. Finding an optimal partition is a difficult task since the set of all possible partitions is too large to be searched for. Instead of searching for the best partition, we will therefore construct partitions which are quasi-optimal in the sense that they can be computed with almost linear costs and allow approximants of logarithmic-linear complexity.

The question whether a block b can be approximated by a matrix of low-rank is usually connected with the underlying problem the matrix A stems from. Since we do not want to restrict ourselves to a specific problem in this part of the book, this dependence will be incorporated into an abstract condition, the so-called **admissibility condition**, which is required to satisfy

(i) if b is admissible, then the singular values of A_b decay exponentially;
(ii) the admissibility condition can be checked for each block $t\times s \in \mathscr{P}(I\times J)$; the required amount of arithmetic operations is $\mathscr{O}(|t|+|s|)$;
(iii) if b is admissible, then any subset $b' \subset b$ is admissible, too.

The following three examples give an idea what this admissibility condition could be.

Example 1.12. Consider the function $f(x,y) = (x+y)^{-1}$ for $x,y>0$. This function is singular for $x=y=0$ only. If the matrix $A \in \mathbb{R}^{n\times n}$ arises from evaluating f at given points $x_i > 0$, $i=1,\dots,n$, i.e.,

$$a_{ij} = f(x_i,x_j), \quad i,j=1,\dots,n,$$

then the condition

$$\min\left\{\frac{\operatorname{diam} X_t}{\operatorname{dist}(0,X_t)}, \frac{\operatorname{diam} X_s}{\operatorname{dist}(0,X_s)}\right\} \le \eta,$$

where $X_t := \{x_i, i\in t\} \subset \mathbb{R}$, leads to exponentially decaying singular values of the block $t\times s$ of A provided that $0<\eta<1$. As usual, we define

$$\operatorname{diam} X = \max_{x,y\in X} |x-y| \quad \text{and} \quad \operatorname{dist}(X,Y) = \inf_{x\in X, y_\in Y} |x-y|$$

for $X,Y \subset \mathbb{R}^d$.

The exponential decay of the singular values is visible from the Taylor expansion of f with respect to $y>0$ about $y_0 := \max X_s$

$$f(x,y) = \sum_{\ell=0}^{\infty} \frac{(y-y_0)^\ell}{\ell!} \partial_y^\ell f(x,y_0) = \sum_{\ell=0}^{k-1} (y_0-y)^\ell (x+y_0)^{-\ell-1} + R_k(x,y), \qquad (1.8)$$

where $R_k(x,y) = (y_0-y)^k (x+\tilde{y}_0)^{-k-1}$ with some $\tilde{y}_0 \in [y,y_0]$. If $y \in X_s$, then

$$|R_k(x,y)| \le (x+\tilde{y}_0)^{-1} \left(\frac{\operatorname{diam} X_s}{\operatorname{dist}(0,X_s)} \right)^k \le (x+y)^{-1} \eta^k$$

provided that $\operatorname{diam} X_s \le \eta \operatorname{dist}(0,X_s)$. Interchanging the roles of x and y, a similar result can be obtained under the condition $\operatorname{diam} X_t \le \eta \operatorname{dist}(0,X_t)$.

According to (1.8), the matrices $U \in \mathbb{R}^{t\times k}$ and $V \in \mathbb{R}^{s\times k}$ defined by

$$u_{i\ell} = (x_i + y_0)^{-\ell-1} \quad \text{and} \quad v_{j\ell} = (y_0 - x_j)^{\ell}, \quad i \in t,\, j \in s, \quad \ell = 1,\dots,k,$$

satisfy

$$|a_{ij} - (UV^T)_{ij}| \le \eta^k |a_{ij}|, \quad i \in t,\, j \in s.$$

Hence, A_{ts} can be approximated by a rank-k matrix UV^T with relative error η^k. The exponential decay of the singular values follows from Remark 1.6.

The condition from the next two examples will be important for the second part of this book.

Example 1.13 (Elliptic problems). In the case of finite element discretizations of elliptic operators each index set t will be associated with a subdomain

$$X_t := \bigcup_{i\in t} X_i,$$

which is the union of supports $X_i \subset \Omega$ of finite element basis functions defined on the computational domain $\Omega = \bigcup_{i\in I} X_i$. For this class of problems a block $t \times s$ will be called admissible if the condition

$$\min\{\operatorname{diam} X_t, \operatorname{diam} X_s\} \le \eta \operatorname{dist}(X_t, X_s) \tag{1.9}$$

is satisfied. Condition (1.9) reflects the fact that the Schwartz kernel $S(x,y)$ of elliptic solution operators is singular for $x = y$ only; see Lemma 3.5 and (4.21). Depending on the parameter $\eta > 0$, the singular values of blocks A_{ts} satisfying condition (1.9) will be shown to decay exponentially.

Checking condition (1.9) is expensive. Especially the computation of $\operatorname{dist}(X_t, X_s)$ requires $\mathcal{O}(|t| \cdot |s|)$ operations. In order to be able to achieve an almost linear overall complexity, we have to replace condition (1.9) by a sufficient condition which can be evaluated with $\mathcal{O}(|t| + |s|)$ operations; cf. requirement (ii) on the admissibility condition. We assume that the X_i, $i \in I$, are polygonal. For $t \subset I$ we set

$$\rho_t := \sup\{|x - m_t|, x \in X_t\},$$

where m_t denotes the centroid of X_t. If for $t \subset I$ and $s \subset J$ it holds that

$$2\min\{\rho_t, \rho_s\} + \eta(\rho_t + \rho_s) \le \eta |m_t - m_s|, \tag{1.10}$$

then $\min\{\operatorname{diam} X_t, \operatorname{diam} X_s\} \le \eta \operatorname{dist}(X_t, X_s)$. This follows from $\operatorname{diam} X_t \le 2\rho_t$ and

$$\operatorname{dist}(X_t, X_s) \ge |m_t - m_s| - \rho_t - \rho_s.$$

The evaluation of condition (1.10) can be done with $\mathcal{O}(|t|+|s|)$ operations.

It will be seen that in some cases it is enough to consider only information that is contained in the matrix and omit the geometric information of the underlying grid.

Example 1.14 (Algebraic admissibility). For a given sparse and invertible matrix $A \in \mathbb{C}^{I\times I}$ let

$$G(A) := \big\{(i,j) \in I \times I : a_{ij} \neq 0 \text{ or } a_{ji} \neq 0\big\} \tag{1.11}$$

be its **matrix graph**. We assume that A is irreducible, i.e., for each pair $(i,j) \in I \times I$ there is a path connecting i and j in the matrix graph. By d_{ij} we denote the minimal length of such paths. For $t,s \subset I$ let

$$\operatorname{dist}(t,s) = \min_{i\in t,\, j\in s} d_{ij} \quad \text{and} \quad \operatorname{diam} t = \max_{i,j\in t} d_{ij} \tag{1.12}$$

A condition that is similar to (1.9) is, for instance,

$$\min\{\operatorname{diam} t, \operatorname{diam} s\} \leq \eta \operatorname{dist}(t,s). \tag{1.13}$$

In Sect. 4.1.3 another and even relaxed condition of this kind will be used for the approximation of inverses of sparse matrices.

Consider the admissibility condition (1.9) from Example 1.13, for instance. It will be seen by arguments similar to those used in Example 1.12 that blocks containing matrix indicies from the set

$$\mathcal{Z} := \{(i,j) \in I \times I : \operatorname{dist}(X_i, X_j) = 0\}$$

cannot be approximated by low-rank matrices because $\mathcal{Z}$ represents the part on which the Schwartz kernel $S(x,y)$ is singular. In order to guarantee that such blocks do not spoil the overall complexity, we have to make sure that they are small. This leads to the following definition.

Definition 1.15. A partition P is called **admissible** if each block $t \times s \in P$ is either admissible or small; i.e., the cardinalities $|t|$ and $|s|$ of t and s satisfy $\min\{|t|,|s|\} \leq n_{\min}$ with a given minimal dimension $n_{\min} \in \mathbb{N}$.

Figure 1.1 shows an admissible partition of the set of matrix indices $I \times I$. Obviously, the diagonal $\{(i,i), i \in I\}$ is a subset of $\mathcal{Z}$. Depending on the problem, $\mathcal{Z}$ may however contain also off-diagonal pairs. This happens, for instance, if the problem is defined in more than one spatial dimension. This partition guarantees that the matrix entries corresponding to each block shown in Fig. 1.1 can be approximated by a matrix of small rank. The rank of blocks containing indices from $\mathcal{Z}$ will obviously be small since their dimensions are small.

The way we will construct admissible partitions is common practice for $\mathcal{H}$-matrices; see [133, 18, 114]. In Sect. 1.4 we will define so-called cluster trees, which are hierarchies of partitions of I and J. Their construction in view of the admissibility condition (1.9) will be described in Sect. 1.4.1. Based on a cluster tree for I, an

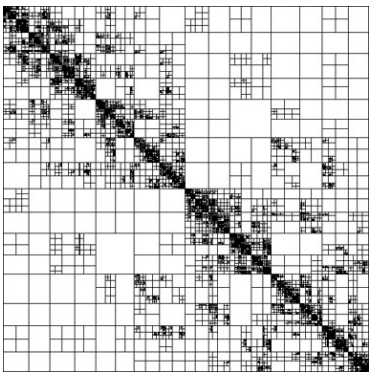


Fig. 1.1 An admissible partition of $I \times I$.

algorithm for partitioning $I \times J$ with a minimum number of blocks is presented in Sect. 1.4.2. This partition is suitable for storing and multiplying A by a vector, but it does not allow to perform matrix operations such as the matrix product and the matrix inverse. If the latter operations are required, the partitioning from Sect. 1.5 has to be used instead. In that section partitions of $I \times J$ for arbitrary admissibility conditions will be constructed based on cluster trees for I and J. Although this will usually not result in an optimal partition, it will satisfy the mentioned requirements. Since the definition of hierarchical matrices will be based on the partitioning from Sect. 1.5, the quality of the partition mainly determines the efficiency of hierarchical matrices. We remark that the purely algebraic method from Sect. 2.6 may be helpful to improve a given partition.

Before we turn to the partitioning of general matrices, the aim of the next section is to see that tensor partitions cannot satisfy the complexity requirements while hierarchical partitions give promising complexities if blocks intersecting the diagonal are assumed to be non-admissible.

1.3.1 Tensor vs. Hierarchical Partitions

In this section we assume that $I = J$. For the comparison of tensor and hierarchical partitions we will investigate the memory consumption and the number of arithmetic operations required to multiply such matrices by a vector if

(i) each diagonal block is stored as a dense matrix;
(ii) all other blocks $t \times s$ are assumed to be admissible and are stored as rank-1 matrices, i.e., $A_{ts} = uv^H$, where $u \in \mathbb{C}^t$ and $v \in \mathbb{C}^s$.

Let us first consider the case of tensor partitions (see Fig. 1.2)

$$P = P_I \times P_I = \{t_k \times t_\ell : t_k, t_\ell \in P_I\},$$

where $P_I := \{t_k, k = 1, \ldots, p\}$ is a partition of the index set I; i.e.,

$$I = \bigcup_{k=1}^{p} t_k \quad \text{and} \quad t_k \cap t_\ell = \varnothing \quad \text{for } k \neq \ell.$$

Storing A requires $\sum_{k=1}^{p} |t_k|^2$ units of storage for the diagonal and

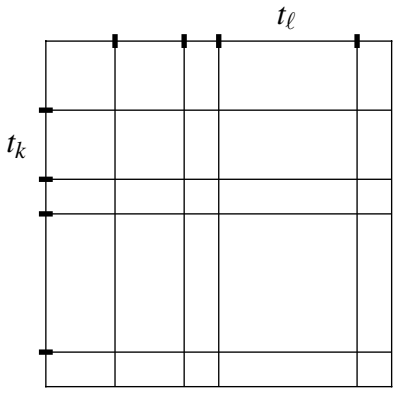


Fig. 1.2 Tensor product partition.

$$\sum_{k \neq \ell} |t_k| + |t_\ell| = 2 \sum_{k=1}^{p} \sum_{\substack{\ell=1 \\ \ell \neq k}}^{p} |t_k| = 2(p-1) \sum_{k=1}^{p} |t_k| = 2(p-1)|I|$$

units for the off-diagonal blocks. Due to the Cauchy-Schwarz inequality

$$|I|^2 = \left(\sum_{k=1}^{p} |t_k| \right)^2 \leq p \sum_{k=1}^{p} |t_k|^2,$$

at least $|I|^2/p + 2(p-1)|I|$ units of storage are necessary to hold A. The minimum of the previous expression is attained for $p = \sqrt{|I|/2}$ resulting in a minimum amount of storage of order $|I|^{3/2}$. Hence, the required amount of storage resulting from tensor product partitions is not competitive.

Let us now check whether a hierarchical partition leads to almost linear complexity. For the ease of notation we restrict ourselves to a number n of unknowns which is a power of 2; i.e., $n = 2^p$ for some $p \in \mathbb{N}$. By the following recursion we will define a special hierarchical partition of a matrix $A \in \mathbb{C}^{I \times I}$. Assume that t has already been generated from I after ℓ subdivisions. Subdividing $t = \{i_1, \ldots, i_{2^{p-\ell}}\}$ into two parts

$$t_1 = \{i_1, \ldots, i_{2^{p-\ell-1}}\} \quad \text{and} \quad t_2 = \left\{i_{2^{p-\ell-1}+1}, \ldots, i_{2^{p-\ell}}\right\}$$

of equal size, we obtain a 2×2 block partition of $A_{tt} \in \mathbb{C}^{t \times t}$:

$$A_{tt} = \begin{bmatrix} A_{t_1 t_1} & A_{t_1 t_2} \\ A_{t_2 t_1} & A_{t_2 t_2} \end{bmatrix}, \tag{1.14}$$

where $A_{t_i t_j} \in \mathbb{C}^{t_i \times t_j}$, $i, j = 1, 2$. Similarly to the case of tensor partitions, we restrict the set of all $A \in \mathbb{C}^{I \times I}$ by the condition that the off-diagonal blocks $A_{t_1 t_2}$ and $A_{t_2 t_1}$ are rank-1 matrices. The diagonal blocks $A_{t_1 t_1}$ and $A_{t_2 t_2}$, however, are subdivided in the same way as A_{tt}; i.e., its off-diagonal blocks are again restricted to rank-1 matrices while its diagonal blocks are subdivided and so on. This recursion stops after p steps when the diagonal blocks are 1×1-blocks and cannot be subdivided any further. The resulting partition in the case $p = 4$ is depicted in Fig. 1.3. The set of such matrices will be denoted by $\mathscr{H}_p$. Since apart from dense blocks on the

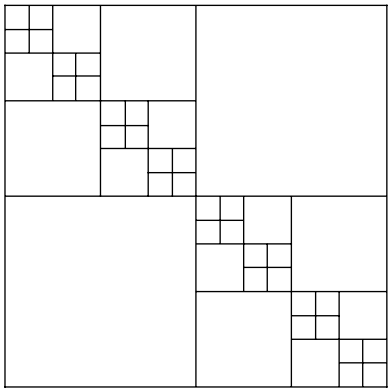


Fig. 1.3 Hierarchical partition.

diagonal, only low-rank blocks will appear in this example, we do not consider the most general case of a hierarchical matrix. However, the structure is general enough to study all important effects.

Let N_p^{st} denote the amount of storage which is required to hold an $\mathscr{H}_p$-matrix. By the following considerations it will be seen that N_p^{st} scales almost linearly with respect to n. Since the off-diagonal blocks in level p require $4 \cdot 2^{p-1} = 2^{p+1}$ units of storage, we find the recursive relation

$$N_p^{\mathrm{st}} = 2N_{p-1}^{\mathrm{st}} + 2^{p+1}$$

with $N_0^{\mathrm{st}} = 1$. Resolving this recursion, we obtain that

$$N_p^{\mathrm{st}} = (2p+1)2^p = 2n \log_2 n + n.$$

Hence, this hierarchical partition satisfies the efficiency requirements that were mentioned in the introduction. Compared with the tensor partition, the expensive diagonal part in the hierarchical partition is significantly smaller.

In addition to storing the matrix, usually also arithmetic operations have to be performed. We consider the matrix-vector product Ax of a matrix $A \in \mathscr{H}_p$ of this special hierarchical structure and a vector $x \in \mathbb{C}^I$. Using the blocking (1.14), this is done recursively by

$$A_{tt}x_t = \begin{bmatrix} A_{t_1t_1}x_{t_1} + A_{t_1t_2}x_{t_2} \\ A_{t_2t_1}x_{t_1} + A_{t_2t_2}x_{t_2} \end{bmatrix},$$

where $x_t = [x_{t_1}^T, x_{t_2}^T]^T$ and $x_{t_1} \in \mathbb{C}^{t_1}$, $x_{t_2} \in \mathbb{C}^{t_2}$. Hence, for evaluating Ax we need the results of the products $A_{t_1t_1}x_{t_1}$ and $A_{t_2t_2}x_{t_2}$, which both are the same kind of problems as $A_{tt}x_t$ but have half the size. If N_p^{MV} denotes the number of arithmetic operations that are required to multiply a $2^p \times 2^p$-matrix $A \in \mathscr{H}_p$ by $x \in \mathbb{C}^n$, then it holds that

$$N_p^{\mathrm{MV}} = 2N_{p-1}^{\mathrm{MV}} + 2^{p+2} - 2,$$

because multiplying the rank-1 matrices $A_{t_1t_2}$ and $A_{t_2t_1}$ by x_{t_2} and x_{t_1} and adding the results each requires $4 \cdot 2^{p-1} - 1$ operations; cf. Sect. 1.1.1. With $N_0^{\mathrm{MV}} = 2$ this relation results in

$$N_p^{\mathrm{MV}} = p2^{p+2} + 2 = 4n\log_2 n + 2.$$

Therefore, hierarchical partitions seem to provide a means to guarantee competitive complexities. If rank-k matrices are used instead of rank-1 matrices, the complexities for storing and multiplying A by a vector are of the order $kn\log_2 n$. Table 1.2 shows that although the hierarchical representation reduces the complexity by almost an order of n, this representation pays compared with the entrywise representation, which requires $2n^2$ operations, only for n which are large enough.

Table 1.2 N_p^{MV} for standard and hierarchical representation.

p	n	$2n^2$	$4kn\log_2 n + 2$	improvement
0	1	2	2	no
1	2	8	$8k+2$	no
2	4	32	$32k+2$	no
3	8	128	$96k+2$	no
4	16	512	$256k+2$	yes iff $k \leq 1$
5	32	2 048	$640k+2$	yes iff $k \leq 3$
6	64	8 192	$1536k+2$	yes iff $k \leq 5$
7	128	32 768	$3584k+2$	yes iff $k \leq 9$

The presented partition is too special since non-admissible blocks can only appear on the diagonal. Such a situation occurs, for instance, for one-dimensional elliptic operators. For higher spatial dimensions non-admissible blocks will also appear in other parts of the matrix. In the following sections it will therefore be

described how hierarchical partitions can be constructed for arbitrary admissibility conditions.

1.4 Cluster Trees

Since searching the whole set of partitions of $I \times J$ for an optimal partition is not an option, we restrict the search to blocks $b = t \times s$ consisting of sets of indices t and s which are generated by recursive subdivision of I and J, respectively. Note that this will usually not lead to an optimal partition. However, it will be shown under realistic assumptions that the resulting partition is good enough in the sense that it leads to almost linear complexity. In order to partition the set of matrix indices $I \times J$ hierarchically into sub-blocks, we first need a rule to subdivide the index sets I and J. This leads to the so-called cluster tree (cf. [138]), which contains a hierarchy of partitions.

Definition 1.16. A tree $T_I = (V, E)$ with vertices V and edges E is called a **cluster tree** for a set $I \subset \mathbb{N}$ if the following conditions hold

(i) I is the root of T_I;
(ii) $\varnothing \neq t = \dot{\bigcup}_{t' \in S(t)} t'$ for all $t \in V \setminus \mathscr{L}(T_I)$;
(iii) the degree $\deg t := |S(t)| \geq 2$ of each vertex $t \in V \setminus \mathscr{L}(T_I)$ is bounded from below.

Here, the set of sons $S(t) := \{t' \in V : (t, t') \in E\}$ of $t \in V$ is pairwise disjoint and

$$\mathscr{L}(T_I) := \{t \in V : S(t) = \varnothing\}$$

denotes the set of leaves of T_I.

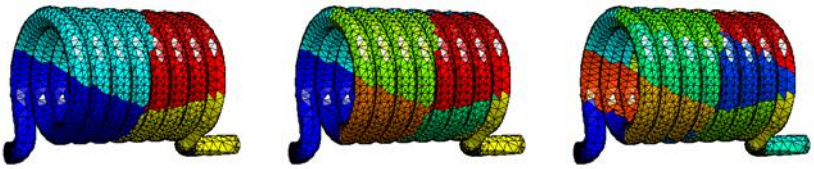
The **level** of $t \in T_I$ is the distance to the root, i.e., the number of applications of S to I that are required to obtain t. Condition (ii) implies that $t \subset I$ for all $t \in T_I$ and that each level

$$T_I^{(\ell)} := \{t \in T_I : \operatorname{level} t = \ell\}$$

of T_I contains a partition of I.

In the sequel we will identify the set of vertices V with the cluster tree T_I. This slight abuse in notation will not be harmful since the tree structure of T_I can always be constructed by recursively applying the mapping $t \mapsto S(t)$ to I. Hence, two cluster trees for I differ only by the mapping S. The actual way clusters are subdivided depends on the application. In Example 1.13 each cluster t was associated with a part of the computational domain. In this case the purpose of S is to generate subdomains of minimal diameter; see Fig. 1.4.

Remark 1.17. For practical purposes it is useful to work with clusters having a minimal size of $n_{\min} > 1$ rather than subdividing the clusters until only one index is left. One reason for this is that the outer-product representation does not pay off



2nd level (4 parts) 3rd level (8 parts) 4th level (16 parts)

Fig. 1.4 Three levels in a cluster tree.

for small blocks; see Definition 1.3. A minimal cluster size $n_{\min} > 1$ will finally lead to a minimal blocksize in the matrix partition. Hence, another reason for the introduction of a minimal cluster size is the exploitation of cache effects on modern computer architectures.

Since the number of leaves $|\mathscr{L}(T_I)|$ is bounded by $|I|/n_{\min}$ provided that $|t| \geq n_{\min}$ for all $t \in T_I$, the following estimate shows that the complexity of storing a cluster tree is still linear. The proof uses the property that each sub-tree of T_I is a cluster tree.

Lemma 1.18. *Let* $q := \min_{t \in T_I \setminus \mathscr{L}(T_I)} \deg t \geq 2$. *Then for the number of vertices in* T_I *it holds that*

$$|T_I| \leq \frac{q|\mathscr{L}(T_I)| - 1}{q-1} \leq 2|\mathscr{L}(T_I)| - 1.$$

Proof. We cut down the tree $T := T_I$ vertex by vertex starting from the leaves of $T \setminus \mathscr{L}(T)$ in k steps until only the root is left. Let T_ℓ denote the tree after ℓ steps and q_ℓ the degree of the ℓth vertex. Then $|T_{\ell+1}| = |T_\ell| - q_\ell$ and $|\mathscr{L}(T_{\ell+1})| = |\mathscr{L}(T_\ell)| - q_\ell + 1$. After k steps $|T_k| = 1 = |\mathscr{L}(T_k)|$, where

$$|T_k| = |T| - \sum_{\ell=1}^{k-1} q_\ell \quad \text{and} \quad |\mathscr{L}(T_k)| = |\mathscr{L}(T)| - \sum_{\ell=1}^{k-1} (q_\ell - 1).$$

Hence, $|T| = |\mathscr{L}(T)| + k - 1$ and from $q_k \geq q$ it follows that $k(q-1) \leq |\mathscr{L}(T)| + q - 2$. □

In order to be able to estimate the asymptotic complexity of algorithms working on trees, we need to consider families of trees parametrized by the cardinality $|I|$ of I. As we have seen in the previous lemma, the number of vertices in T_I is always linear. For a logarithmic depth of T_I we have to ensure that each subdivision by S produces clusters of comparable cardinality.

Definition 1.19. A tree T_I is called **balanced** if

$$R := \min_{t \in T_I \setminus \mathscr{L}(T_I)} \{|t_1|/|t_2|, t_1, t_2 \in S(t)\}$$

is bounded independently of $|I|$ by a positive constant from below.

Figure 1.5 shows two cluster trees for $I := \{1,\dots,n\}$ each consisting of three levels. The left tree is balanced in the sense of the previous definition since $R = 2$, while the right tree is unbalanced since for this tree $R = 1/(|I|-1)$.

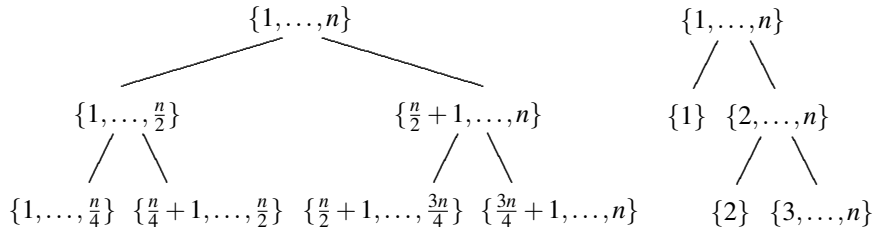


Fig. 1.5 A balanced and an unbalanced cluster tree.

By $L(T_I) := \max_{t\in T_I} \operatorname{level} t + 1$ we denote the **depth** of the cluster tree T_I. The depth of balanced cluster trees depends logarithmically on $|I|$.

Lemma 1.20. *Let T_I be a balanced cluster tree with $q := \min_{t\in T_I\setminus\mathscr{L}(T_I)} \deg t \geq 2$. Then for the depth of T_I it holds that*

$$L(T_I) \leq \log_\xi(|I|/n_{\min}) + 1 \sim \log|I|$$

and $|t| \leq |I|\xi^{-\ell}$, where ℓ denotes the level of $t \in T_I$ and $\xi := R(q-1)+1$.

Proof. For $t \in T_I \setminus \mathscr{L}(T_I)$ and $t' \in S(t)$ we observe that

$$\begin{aligned}\frac{|t|}{|t'|} = \frac{|t'| + \sum_{t'\neq s\in S(t)} |s|}{|t'|} &= 1 + \sum_{t'\neq s\in S(t)} \frac{|s|}{|t'|} \geq 1 + (|S(t)|-1)R \\ &\geq 1 + (q-1)R = \xi.\end{aligned}$$

Let $e_1,\dots,e_{L-1}$ be a sequence of edges from the root $v_1 := I$ to a deepest vertex v_L in T_I. Furthermore, let $v_2,\dots,v_{L-1}$ be the intermediate vertices. Then from

$$\xi|v_{\ell+1}| \leq |v_\ell|, \quad \ell = 1,\dots,L-1,$$

we obtain that

$$\xi^{L-1}|v_L| \leq |I|,$$

which gives $(L-1)\log\xi \leq \log(|I|/|v_L|) \leq \log(|I|/n_{\min})$. □

The following lemma will be helpful for many complexity estimates in the rest of this chapter.

Lemma 1.21. *Let T_I be a cluster tree for I, then*

$$\sum_{t\in T_I} |t| \leq L(T_I)|I| \quad \textit{and} \quad \sum_{t\in T_I} |t|\log|t| \leq L(T_I)|I|\log|I|.$$

Furthermore, for any $c > 0$ it holds that

$$\sum_{t \in T_I} \min\{c, |t|^2\} \leq 3\sqrt{c}|I|. \tag{1.15}$$

Proof. Each of the $L(T_I)$ levels in T_I consists of disjoint subsets of I. The second estimate follows from $\log|t| \leq \log|I|$ for $t \in T_I$.

For the proof of (1.15) we restrict ourselves to cluster trees with constant degree $\deg t = q \geq 2$ and $|t| = |I|q^{-\ell}$ for all $t \in T_I^{(\ell)}$, $\ell = 0, \ldots, L(T_I) - 1$. If $n_{\min} \geq \sqrt{c}$, then from Lemma 1.18 it follows that

$$\sum_{t \in T_I} \min\{c, |t|^2\} \leq \sum_{t \in T_I} c = c\,|T_I| \leq \frac{q}{q-1}\frac{c}{n_{\min}}|I| \leq \frac{q}{q-1}\sqrt{c}|I|. \tag{1.16}$$

If, on the other hand, $n_{\min} < \sqrt{c}$, then let $\ell_c \in \{0, \ldots, L(T_I) - 1\}$ be the largest index such that $|t| \geq \sqrt{c}$ for all $t \in T_I^{(\ell)}$, $\ell \leq \ell_c$. Estimate (1.16) applied to this part of the cluster tree gives

$$\sum_{\ell=0}^{\ell_c} \sum_{t \in T_I^{(\ell)}} \min\{c, |t|^2\} \leq \frac{q}{q-1}\sqrt{c}|I|,$$

while

$$\sum_{\ell=\ell_c+1}^{L(T_I)-1} \sum_{t \in T_I^{(\ell)}} \min\{c, |t|^2\} = \sum_{\ell=\ell_c+1}^{L(T_I)-1} \sum_{t \in T_I^{(\ell)}} |t|^2 = \sum_{\ell=\ell_c+1}^{L(T_I)-1} q^{\ell}|I|^2 q^{-2\ell} \leq \frac{|I|^2}{q-1} q^{-\ell_c - 1}.$$

The assertion follows from $|t| = |I|q^{-\ell_c - 1} < \sqrt{c}$ for all $t \in T_I^{(\ell_c+1)}$. □

We have seen that the number of vertices in T_I scales linearly with respect to $|I|$. For each vertex t in T_I its indices have to be stored. Hence, by Lemma 1.21 the amount of storage for a balanced tree T_I is of the order $|I| \log |I|$.

It is desirable that clusters are contiguous. For this purpose, the set I should be reordered. If t is to be subdivided into t_1 and t_2, we rearrange the indices in t so that $\max t_1 \leq \min t_2$. This can be done during the construction of T_I. In this case a cluster t can be represented by its minimal and maximal index. With this simplification, T_I requires $|T_I| \sim |I|$ units of storage even for unbalanced trees. The permutation requires additional $|I|$ units of storage. Note that due to the nested structure of cluster trees, this reordering does not change the previously generated clusters.

Remark 1.22. If the indices are reordered during subdivision, we have only two possibilities, $t = [t_1, t_2]$ or $t = [t_2, t_1]$, for arranging the indices of t satisfying $\deg t = 2$; i.e., the usage of cluster trees leaves room for only $2^{L-1} n_{\min}!$ permutations of I. Here, we assume that the size of the leaves in T_I is exactly $n_{\min}$. Hence, building a binary cluster tree determines the numbering of the indices in I up to $\mathcal{O}(n)$ permutations.

1.4.1 Construction of Cluster Trees

A cluster t is subdivided for the purpose of refinement. A refined cluster pair and its sub-blocks may be admissible while the father pair is not. The latter principle is reflected by the required property (iii) of the admissibility condition from Sect. 1.3. Hence, a certain aim which depends on the admissibility condition and therefore on the application has to be kept in mind when subdividing t. The following section is hence divided into two parts. In the first part we will present clustering algorithms for the case that geometric information is associated with the indices. The second part treats the construction of cluster trees based on the matrix graph of a sparse matrix.

1.4.1.1 Geometric Clustering

In the case of finite element discretizations of elliptic operators each cluster t is associated with the union

$$X_t := \bigcup_{i \in t} X_i,$$

where X_i denotes a part of the computational domain $\Omega \subset \mathbb{R}^d$; see Example 1.13. In this case subdividing t has the purpose of reducing the diameter of X_t. Hence, in this section we will concentrate on how a cluster tree T_I is constructed from an index set $I \subset \mathbb{N}$ such that the diameters of X_t are as small as possible. For other applications the purpose of subdivision may be different. In this case the following algorithm has to be replaced by an appropriate one.

In this section we assume that X_i, $i \in I$, are **quasi-uniform**, i.e., there is a constant $c_U > 0$ such that

$$\max_{i \in I} \mu(X_i) \leq c_U \min_{i \in I} \mu(X_i), \tag{1.17}$$

and **shape regular**, i.e.,

$$\mu(X_i) \geq c_R (\operatorname{diam} X_i)^m, \quad i \in I, \tag{1.18}$$

with some constant $c_R > 0$. The expression $\mu(M)$ denotes the measure of an m-dimensional manifold $M \subset \mathbb{R}^d$, i.e., the area if M is $(d-1)$-dimensional and the volume if M is d-dimensional. We assume that the computational domain Ω is an m-dimensional manifold, i.e., there is a constant $c_\Omega > 0$ such that for all $z \in \Omega$

$$\mu(\Omega \cap B_r(z)) \leq c_\Omega r^m \quad \text{for all } r > 0. \tag{1.19}$$

In the case $m = d$, c_Ω coincides with the volume ω_d of the unit ball in $\mathbb{R}^d$. For $m = d-1$ the constant c_Ω will depend on the curvature of the hypersurface $\Omega \subset \mathbb{R}^d$. In addition, we assume that only a bounded number of sets X_i overlap; i.e., there is $\nu \in \mathbb{N}$ such that for each $i \in I$

$$\max_{x \in \operatorname{int} X_i} |\{j \in I : x \in \operatorname{int} X_j\}| \leq \nu. \tag{1.20}$$

The above assumptions are in accordance with usual applications such as finite element discretizations.

Due to Definition 1.16, for the construction of T_I we only have to devise a procedure which divides a cluster t appropriately. An obvious approach (cf. [103]) is to find an axial parallel box Q which contains X_t. By recursively dividing Q into 2^d sub-cubes of equal size in each step, one obtains a hierarchy of decompositions of I. The disadvantage of this procedure is that no information about the shape of X_t is used for its subdivision. As a consequence, branches of the tree may be empty. This situation occurs, for instance, in applications where the data is clustered on low-dimensional manifolds. Then the cluster tree is likely to be unbalanced; see Fig. 1.6.

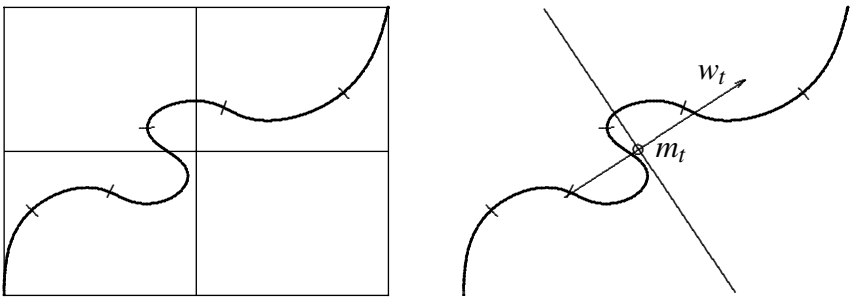


Fig. 1.6 Bounding box and PCA subdivision.

We favor the following alternative clustering strategy which is based on the **principal component analysis** (PCA) that provides well-balanced cluster trees. This method was originally designed to find the main direction of a set of points; cf. [201]. In order to be able to apply it, we select arbitrary but fixed points $z_i \in X_i$, $i = 1, \ldots, n$, e.g., the centroids of X_i if X_i are polygonal.

A cluster $t \subset I$ is subdivided by the hypersurface through the centroid

$$m_t := \frac{\sum_{i \in t} \mu(X_i) z_i}{\sum_{i \in t} \mu(X_i)}$$

of t with normal w_t, where w_t is the main direction of t.

Definition 1.23. A vector $w \in \mathbb{R}^d$, $\|w\|_2 = 1$, satisfying

$$\sum_{i \in t} |w^T (z_i - m_t)|^2 = \max_{\|v\|_2 = 1} \sum_{i \in t} |v^T (z_i - m_t)|^2$$

is called a **main direction** of the cluster t.

Note that with the symmetric positive semi-definite covariance matrix

$$C_t := \sum_{i \in t} (z_i - m_t)(z_i - m_t)^T \in \mathbb{R}^{d \times d}$$

it holds that

$$\sum_{i \in t} |v^T (z_i - m_t)|^2 = \sum_{i \in t} v^T (z_i - m_t)(z_i - m_t)^T v = v^T C_t v \quad \text{for all } v \in \mathbb{R}^d.$$

Hence, by the variational representation of eigenvalues

$$\max_{\|v\|_2 = 1} \sum_{i \in t} |v^T (z_i - m_t)|^2 = \max_{\|v\|_2 = 1} v^T C_t v = \lambda_{\max}(C_t)$$

one observes that the maximum is attained for the eigenvector corresponding to the largest eigenvalue $\lambda_{\max}$ of C_t. The power method (cf. [148]) is well-suited for the computation of the main direction since it converges the faster the more the extensions of X_t differ.

Using w_t, one possibility is to define the sons $S(t) = \{t_1, t_2\}$ of t by

$$t_1 = \{i \in t : w_t^T (z_i - m_t) > 0\} \tag{1.21}$$

and $t_2 := t \setminus t_1$. This subdivision generates **geometrically balanced** cluster trees in the sense that there are constants $c_g, c_G > 0$ such that for each level $\ell = 0, \ldots, L(T_I) - 1$

$$(\operatorname{diam} X_t)^m \le c_g 2^{-\ell} \quad \text{and} \quad \mu(X_t) \ge 2^{-\ell} / c_G \quad \text{for all } t \in T_I^{(\ell)}. \tag{1.22}$$

Geometrically balanced cluster trees will be important later on for the complexity of admissible partitions; see Example 1.36.

Another possibility is to reorder the set t by sorting the indices with respect to the sizes of their projections $w_t^T (z_i - m_t)$, i.e., by finding a bijection $\pi : \{1, \ldots, |t|\} \to t$ such that

$$w_t^T z_{\pi(i)} \le w_t^T z_{\pi(j)} \quad \text{for } i < j, \quad i, j = 1, \ldots, |t|.$$

Sorting a set of $|t|$ elements can be done with $\mathcal{O}(|t| \log |t|)$ operations. Define $t_1 \subset t$ by

$$t_1 = \{\pi(i), i = 1, \ldots, \lceil |t|/2 \rceil\}. \tag{1.23}$$

and t_2 as the second half of t. This construction of course leads to a balanced tree (in the sense of Definition 1.19 with $R = 1$), but (1.22) will not hold in general. In the non-uniform case, these two properties cannot be satisfied in parallel (see Example 1.28) such that one has to concentrate on a balanced tree in order to guarantee a competitive overall complexity; cf. [131]. This problem can be solved by the the algebraic clustering from the second part of this section.

An important property of both subdivisions (1.21) and (1.23) is that always the largest extension of X_t is reduced. As a consequence, the generated clusters will not degenerate. To be more precise, we make the following considerations. Assume that a cluster X_t of level ℓ is enclosed in a cuboid which is aligned with the orthogonal eigenvectors of C_t with non-increasingly ordered side lengths $a_1^{(\ell)} \ge \ldots \ge a_d^{(\ell)}$. Let X_{t_1} and X_{t_2} be generated using either subdivision from above; i.e., each time two

parts are generated by dividing the longest side length $a_1^{(\ell)}$ of the cuboid at a ratio of $0 < \lambda < 1$. Since X_t is subdivided only with respect to one direction, $d-1$ of the side lengths will stay the same. Depending on which the largest side length after the subdivision is, three cases have to be distinguished:

$$\frac{a_1^{(\ell+1)}}{a_d^{(\ell+1)}} = \begin{cases} \frac{\lambda a_1^{(\ell)}}{a_d^{(\ell)}} \leq \frac{a_1^{(\ell)}}{a_d^{(\ell)}}, & \text{if } \lambda a_1^{(\ell)} = a_1^{(\ell+1)}, \\ \frac{a_2^{(\ell)}}{\lambda a_1^{(\ell)}} \leq \lambda^{-1}, & \text{if } \lambda a_1^{(\ell)} = a_d^{(\ell+1)}, \\ \frac{a_2^{(\ell)}}{a_d^{(\ell)}} \leq \frac{a_1^{(\ell)}}{a_d^{(\ell)}}, & \text{else.} \end{cases} \tag{1.24}$$

Hence, the smallest and the largest side length of X_{t_1} differ by a factor of at most

$$\max(\lambda^{-1}, a_1^{(\ell)}/a_d^{(\ell)}).$$

The same ratio for X_{t_2} is bounded by $\max((1-\lambda)^{-1}, a_1^{(\ell)}/a_d^{(\ell)})$.

It is easy to see that quasi-uniformity (1.17) and assumption (1.20) lead to an equivalence of the measure of a cluster and the number of contained indices.

Lemma 1.24. *It holds that* $|t|/(c_U \nu) \leq \mu(X_t)\,(\max_{i\in I} \mu(X_i))^{-1} \leq |t|$ *for all* $t \subset I$.

Proof. It is obvious that

$$\mu(X_t) \leq \sum_{i\in t} \mu(X_i) \leq |t| \max_{i\in I} \mu(X_i).$$

On the other hand, (1.20) and (1.17) give $\nu\mu(X_t) \geq \sum_{i\in t} \mu(X_i) \geq |t| \max_{i\in I} \mu(X_i)/c_U$. □

Cardinality balanced clustering leads to geometrically balanced trees under suitable assumptions.

Theorem 1.25. *Assume that* X_i, $i \in I$, *are quasi-uniform. Then the construction of a cardinality balanced cluster tree using* (1.23) *results in a geometrically balanced tree; i.e.,* (1.22) *is satisfied.*

Proof. We assume that $|I| = 2^p$ for some $p \in \mathbb{N}$. Since $|t| = 2^{-\ell}|I|$ for $t \in T_I^{(\ell)}$, we obtain from Lemma 1.24 that

$$\mu(X_t) \geq |t| \max_{i\in I} \mu(X_i)/(c_U\nu) = 2^{-\ell}|I| \max_{i\in I} \mu(X_i)/(c_U\nu) \geq 2^{-\ell}\mu(\Omega)/(c_U\nu),$$

since $\mu(\Omega) \leq \sum_{i\in I} \mu(X_i) \leq |I| \max_{i\in I} \mu(X_i)$. On the other hand,

$$\mu(X_t) \leq |t| \max_{i\in I} \mu(X_i) = 2^{-\ell}|I| \max_{i\in I} \mu(X_i) \leq c_U\nu 2^{-\ell}\mu(\Omega),$$

because $\nu\mu(\Omega) \geq \sum_{i\in I} \mu(X_i) \geq |I| \min_{i\in I} \mu(X_i)$. Hence, according to (1.18) for the minimal extension δ of X_t it holds that

$$\delta^m \le \mu(X_t)/c_R \le c_U \nu 2^{-\ell} \mu(\Omega)/c_R. \tag{1.25}$$

In order to find a bound for the largest extension, we have to estimate the size of λ from (1.24). A detailed analysis for general clusters X_t is cumbersome. We restrict ourselves to the case of a cuboid Ω with side length $a_1 \ge \ldots \ge a_d$. Since

$$\lambda \mu(X_t) = \mu(X_{t_1}) \ge 2^{-(\ell+1)} \mu(\Omega)/(c_U \nu)$$

and $\mu(X_t) \le c_U \nu 2^{-\ell} \mu(\Omega)$, we find

$$\lambda \ge \frac{1}{2}(c_U \nu)^{-2}.$$

Due to (1.24), the smallest and the largest side length of X_{t_1} differ by a factor of at most $c := \max(\lambda^{-1}, a_1/a_d) \le \max(2(c_U \nu)^2, a_1/a_d)$. We obtain from (1.25) that

$$(\operatorname{diam} X_t)^m \le c^m c_U \nu 2^{-\ell} \mu(\Omega)/c_R,$$

which proves the assertion. □

The following lemma shows that geometrically balanced cluster trees are balanced in the case of quasi-uniform grids. Cardinality and geometrically balanced clustering are therefore equivalent for this kind of grids.

Lemma 1.26. *Assume that X_i, $i \in I$, are quasi-uniform. Then a geometrically balanced cluster tree is balanced in the sense of Definition 1.19.*

Proof. Let $t_1, t_2 \in T_I^{(\ell)}$ be two clusters from the same level ℓ of T_I. From (1.20), (1.19), and (1.22) we see that

$$|t_1| \min_{i \in t_1} \mu(X_i) \le \sum_{i \in t_1} \mu(X_i) \le \nu \mu(X_{t_1}) \le \nu c_\Omega (\operatorname{diam} X_{t_1})^m \le \nu c_\Omega c_g 2^{-\ell}.$$

The previous estimate together with

$$|t_2| \max_{i \in t_2} \mu(X_i) \ge \sum_{i \in t_2} \mu(X_i) \ge \mu(X_{t_2}) \ge 2^{-\ell}/c_G$$

leads to

$$\frac{|t_1|}{|t_2|} \le \nu c_\Omega c_g c_G \frac{\max_{i \in t_2} \mu(X_i)}{\min_{i \in t_1} \mu(X_i)} \le \nu c_\Omega c_g c_G c_U,$$

which proves the assertion. □

Since subdividing $t \subset I$ requires $\mathcal{O}(|t|)$ operations for geometrically balanced clustering (1.21) and $\mathcal{O}(|t| \log |t|)$ for cardinality balanced clustering (1.23), we can apply Lemma 1.21 to see that constructing T_I takes $L(T_I)|I|$ and $L(T_I)|I| \log |I|$ operations, respectively. For quasi-uniform polyhedrizations the cluster trees will be balanced. The following theorem is a consequence of Lemma 1.20.

Theorem 1.27. *The construction of T_I using* (1.21) *or* (1.23) *requires $\mathcal{O}(|I|\log|I|)$ operations for geometrically balanced clustering and $\mathcal{O}(|I|\log^2|I|)$ for cardinality balanced clustering. The resulting cluster tree T_I is both balanced and geometrically balanced provided that the sets X_i, $i \in I$, are quasi-uniform. T_I can be stored with complexity $\mathcal{O}(|I|)$.*

The assumption that the sets X_i are quasi-uniform is essential if geometrically balanced clusters are to be generated. This can be seen from the following one-dimensional example.

Example 1.28. Let $X_i = [\frac{2}{3}2^{-i}, \frac{4}{3}2^{-i}]$ with midpoints $z_i = 2^{-i}$, $i \in t := \{1,\dots,n\}$. This kind of grid results, for instance, from adaptive refinement towards a singularity at 0; see Fig. 1.7.

Fig. 1.7 Adaptive refinement.

If we subdivide this set into $t_1 = \{i \in t : z_i < m_t\}$ and $t_2 = t \setminus t_1$, where

$$m_t := \frac{1}{n}\sum_{i\in t} z_i = \frac{1}{n}\sum_{i\in t} 2^{-i} = \frac{1}{n}(1-2^{-n}),$$

then for $i \in t_1$ it holds that $n2^{-i} < 1-2^{-n}$, which is satisfied for all $i > \log_2 2n$. The cluster tree will not be balanced, since t_1 contains most of the indices while t_2 has only few of them.

This situation becomes even worse if we use the centroid

$$m_t := \frac{\sum_{i=1}^n \mu(X_i) z_i}{\sum_{i=1}^n \mu(X_i)}$$

with $\mu(X_i) = \frac{2}{3}2^{-i}$. From

$$m_t = \frac{\sum_{i=1}^n 4^{-i}}{\sum_{i=1}^n 2^{-i}} = \frac{1}{3}\frac{1-4^{-n}}{1-2^{-n}}$$

we see that $i \in t_1$ is equivalent to $3\cdot 2^{-i}(1-2^{-n}) < 1-4^{-n}$, which is satisfied for all $i > \log_2 3$.

1.4.1.2 A Construction Based on Aggregation

In contrast to the previous construction, which relies on subdivision, the following idea is based on aggregation of clusters and constructs the cluster tree from its bottom. Again, we denote by $z_i \in X_i$, $i = 1,\dots,n$, points which represent centers of the

sets X_i. In order to generate $k := n/n_{\min}$ clusters on the bottom level of T_I, we solve the so-called k**-center problem**, i.e., given a set of n points, find a partition into clusters $t_1, \ldots, t_k$ with centers $c_1, \ldots, c_k$ so as to minimize the cost functional

$$\max_{i=1,\ldots,k} \max_{j \in t_i} \|z_j - c_i\|_2.$$

This problem is known to be NP-hard; see [39]. The **farthest-point clustering** algorithm (see [107]) computes an approximation with complexity $\mathcal{O}(nk)$; see [90] for a two-phase algorithm with optimal complexity $\mathcal{O}(n \log k)$. The farthest-point clustering algorithm chooses an arbitrary initial center c_1. In step $i = 1, \ldots, k-1$ find c_{i+1} such that

$$d_i(c_{i+1}) = \max_{j=1,\ldots,n} d_i(z_j),$$

where $d_i(z) := \min_{j \le i} \|z - c_j\|$ denotes the distance of z to the computed centers c_j, $j \le i$. After the computation of $c_1, \ldots, c_k$, assign each point z_i, $i = 1, \ldots, n$, to its nearest center.

Lemma 1.29. *The farthest-point clustering algorithm computes a partition with maximum radius at most twice the optimum.*

Proof. Let c_{k+1} denote the next center chosen in the farthest-point clustering algorithm. Then the maximum radius of the farthest-point clustering solution equals $d_k(c_{k+1})$. Two of the centers $c_1, \ldots, c_{k+1}$, say c_i and c_j with $i < j$, must be in the same cluster with center c of an optimum k-center solution. Since $d_k(c_{k+1}) \le d_{j-1}(c_{k+1})$ and $d_{j-1}(c_{k+1}) \le d_{j-1}(c_j)$, it follows that

$$d_k(c_{k+1}) \le d_{j-1}(c_j) \le \|c_i - c_j\|_2 \le \|c_i - c\|_2 + \|c - c_j\|_2 \le 2\rho,$$

where ρ denotes the maximum radius of an optimum k-center solution. □

Higher levels in the cluster tree can be constructed by clustering the respective centers $c_1, \ldots, c_k$.

1.4.1.3 Algebraic Clustering

In Example 1.38 it will be seen that we can sometimes do without geometrically balanced clusters. The algebraic admissibility condition (1.13) does not involve any information about the underlying geometry. In this case t will be subdivided by partitioning the matrix graph $G(A_{tt}) = (t, E_t)$ of the restriction A_{tt} of a sparse matrix A.

The bipartition can be computed using **spectral bisection** [205] based on the **Fiedler vector**, which is the eigenvector to the second smallest eigenvalue (see [91, 92]) of the Laplace matrix $L_{G(A_{tt})}$ of $G(A_{tt})$ defined as

$$\ell_{ij} = \begin{cases} -1, & \text{if } (i,j) \in E_t, \\ \operatorname{degree}(i), & \text{if } i = j, \end{cases}$$

where $\operatorname{degree}(i) := |\{j \in t \setminus \{i\} : (i,j) \in G(A_{tt})\}|$. Since computing eigenvectors of large matrices is computationally expensive, multi-level ideas (cf. [154]) have been presented to accelerate the process. For this purpose the graph $G(A_{tt})$ is coarsened into a sequence $G^{(1)}, \ldots, G^{(k)}$ such that $|t| > |t^{(1)}| > \cdots > |t^{(k)}|$. Given a graph $G^{(i)} = (t^{(i)}, E^{(i)})$, the next coarser graph $G^{(i+1)}$ is constructed by finding a matching of $G^{(i)}$ and collapsing the matched vertices into multinodes. This is often done by **heavy edge matching** [154], which can be obtained with $\mathcal{O}(|E^{(i)}|)$ operations. Spectral bisection can then be applied to the smallest graph $G^{(k)}$. The resulting partition $P^{(k)}$ is projected back to $G(A_{tt})$ by going through the intermediate partitions $P^{(k-1)}, \ldots, P^{(1)}$. The partition $P^{(i)}$ is obtained from $P^{(i+1)}$ by assigning the vertices $t^{(i)}(v)$ to the part of $P^{(i+1)}$ which v belongs to. The partition $P^{(i+1)}$ can be improved by refinement heuristics such as the **Kernighan-Lin algorithm** [156].

As a result of spectral bisection, each of the resulting parts t_1 and t_2 will contain approximately half of the vertices of t, i.e., we may assume that there are constants $c_b, C_b > 0$ such that

$$c_b 2^{-\ell} |I| \leq |t| \leq C_b 2^{-\ell} |I| \tag{1.26}$$

holds for $t \in T_I^{(\ell)}$, which implies that the resulting cluster tree is balanced. Furthermore, we assume that the diameter of a generated cluster is equivalent to its cardinality in the sense that there are constants $m, c'_d, c''_d > 0$ such that

$$c'_d |t| \leq (\operatorname{diam} t)^m \leq c''_d |t| \quad \text{for all } t \in T_I. \tag{1.27}$$

Due to (1.26) the diameters of two clusters $t \in T_I^{(\ell)}$ and $t' \in T_I^{(\ell')}$ from the ℓth and the ℓ'th level of the cluster tree T_I are comparable; i.e.,

$$(\operatorname{diam} t)^m \leq c_u 2^{\ell' - \ell} (\operatorname{diam} t')^m, \tag{1.28}$$

where $c_u := (c''_d C_B)/(c'_d c_b)$.

The advantage of subdividing clusters t in this manner is that it in some sense minimizes the **edge cut**, i.e. a set of edges $C \subset E_t$ such that $G'_t = (t, E_t \setminus C)$ is no longer connected.

Remark 1.30. Sometimes (see Sect. 4.5) it is also useful to subdivide t into a "left", a "right", and an interface cluster; see Fig. 1.8. The resulting cluster tree will then be ternary. This clustering is closely related with **nested dissection** (see [99, 144]), which is commonly used in domain decomposition methods.

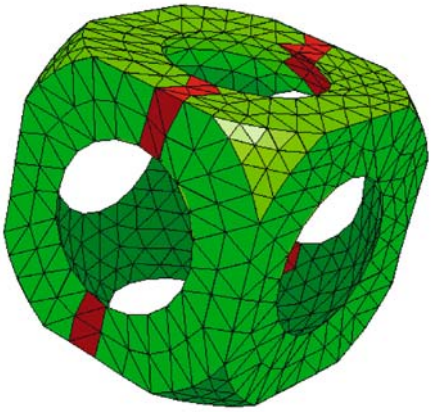


Fig. 1.8 Computational domain with an interface cluster.

1.4.2 Example: An Easily Analyzed Partition

Once the cluster tree T_I for the index set I has been computed, an admissible partition P of $I \times J$ can be constructed from it. The partition from this section achieves a minimal number of blocks since the generated admissible blocks have maximum possible size. In this example we use the admissibility condition

$$\operatorname{diam} X_t \leq \eta \operatorname{dist}(X_t, X_s) \tag{1.29}$$

on the block $t \times s$, which is similar to the condition from Example 1.13.

The basic idea of the following algorithm is to subdivide a given block $t \times s$ into a maximum admissible part $t \times \tau$ and a non-admissible part $t \times (s \setminus \tau)$, where $\tau := \mathscr{F}(t) \cap s$ and

$$\mathscr{F}(t) := \left\{ j \in J : \operatorname{diam} X_t \leq \eta \operatorname{dist}(X_t, X_j) \right\}$$

denotes the **far-field** of $t \subset I$. While $t \times \tau$ fulfills (1.29) and hence is stored in P, the remaining part $t \times (s \setminus \tau)$ of $t \times s$ does not satisfy (1.29) and the procedure is recursively applied to its sub-blocks $t' \times (s \setminus \tau)$, $t' \in S(t)$. If $S(t) = \varnothing$, the recursion stops by adding $t \times (s \setminus \tau)$ as a non-admissible but small block to P.

Remark 1.31. The computation of $\mathscr{F}(t) \cap s$ requires $\mathscr{O}(|t| \cdot |s|)$ operations, which would lead to a quadratic overall complexity. Hence, we replace $\mathscr{F}(t), t \in T_I$, by the set

$$\tilde{\mathscr{F}}(t) := \left\{ j \in J : \operatorname{diam} X_t \leq \frac{\eta}{1+\eta} \operatorname{dist}(z_t, X_j) \right\}, \tag{1.30}$$

where $z_t \in \mathbb{R}^d$ is an arbitrary but fixed point in X_t. It can be easily seen that $\tilde{\mathscr{F}}(t) \subset \mathscr{F}(t)$. The computation of $\tilde{\mathscr{F}}(t) \cap s$ can be performed with $\mathscr{O}(|t| + |s|)$ operations.

Altogether we obtain the following algorithm.

procedure Partition$(t, s, \mathrm{var}\, P)$
Compute $\tau := \tilde{\mathscr{F}}(t) \cap s$
if $\tau \neq \varnothing$ **then** $P := P \cup \{t \times \tau\}$
if $t \in \mathscr{L}(T_I)$ or $|s \setminus \tau| \leq n_{\min}$ **then**
    $P := P \cup \{t \times (s \setminus \tau)\}$
else
    for $t' \in S(t)$ **do** Partition$(t', s \setminus \tau, P)$

Algorithm 1.2: Maximum admissible partition.

It is obvious that applying Algorithm 1.2 to $I \times J$ results in an admissible partition P if P was previously initialized by $P = \varnothing$. Notice that there is no need to construct the cluster tree T_I in advance, since the partitioning algorithm only descends it, which can be simulated by applying S without storing the tree.

The aim of the rest of this section is to estimate the computational cost of Algorithm 1.2 in the case of quasi-uniform grids. Since for each cluster $t \in T_I$ at most the blocks $t \times \tau$ and $t \times (s \setminus \tau)$ are added to P, we immediately obtain that the maximum number

$$c_{\mathrm{sp}}(P) := \max_{t \in T_I} |\{s \subset J : t \times s \in P\}|$$

of blocks in P belonging to a given set of row indices t is bounded by 2. The previous expression will later be generalized to the so-called **sparsity constant** c_{sp}.

Lemma 1.32. *The number $|P|$ of generated blocks is bounded by $2|T_I| \sim |I|$.*

Proof. Since $c_{\mathrm{sp}}(P)$ is bounded by 2, we observe that

$$|P| = \sum_{t \in T_I} |\{s \subset J : t \times s \in P\}| \leq c_{\mathrm{sp}}(P)|T_I| \leq 2|T_I|.$$

The assertion follows from Lemma 1.18. □

In order to be able to estimate the complexity of constructing P, we first have to find a bound for the size of the **near field** $\tilde{\mathscr{N}}(t) := J \setminus \tilde{\mathscr{F}}(t)$. We assume that the sets X_i, $i \in I$, are quasi-uniform and shape regular; i.e.,

$$\max_{i \in I} \operatorname{diam} X_i \leq c_S \min_{i \in I} \operatorname{diam} X_i; \tag{1.31}$$

see (1.17) and (1.18).

Lemma 1.33. *Let T_I be a geometrically balanced cluster tree, i.e.* (1.22) *holds. Under the above assumptions $|\tilde{\mathscr{N}}(t)| \leq c(1+1/\eta)^m |t|$ holds for all $t \in T_I$.*

Proof. From (1.31) it follows that $\operatorname{diam} X_j \leq c_S \operatorname{diam} X_t$, $j \in J$. Since due to (1.30)

$$\operatorname{dist}(z_t, X_j) < (1+1/\eta) \operatorname{diam} X_t$$

for $j \in \tilde{\mathscr{N}}(t)$, the ball $B_\rho(z_t)$ with $\rho := (c_S + 1 + 1/\eta) \operatorname{diam} X_t$ covers $X_{\tilde{\mathscr{N}}(t)}$. Hence, according to (1.19)

$$\mu(X_{\tilde{\mathscr{N}}(t)}) \leq c_\Omega \rho^m = c_\Omega (c_S + 1 + 1/\eta)^m (\operatorname{diam} X_t)^m.$$

Using (1.22), we obtain

$$\mu(X_{\tilde{\mathcal{N}}(t)}) \leq c_g c_G c_\Omega (c_S + 1 + 1/\eta)^m \mu(X_t).$$

Lemma 1.24 leads to the desired estimate. □

Theorem 1.34. *The number of operations and the amount of storage needed to apply Algorithm 1.2 to $I \times J$ is of the order $\eta^{-m}|I|\log|I|$.*

Proof. Algorithm 1.2 descends the cluster tree T_I. In each node $t \in T_I$ at most

$$c(|t| + |\tilde{\mathcal{N}}(t)|)$$

operations are needed since Algorithm 1.2 is applied only to blocks $t \times s$ for which $s \subset \tilde{\mathcal{N}}(t)$ is valid. Lemma 1.21 together with Lemma 1.33 finishes the proof. □

The advantage of the previous partition P is that the number of contained blocks does not depend on η. Blocks have the largest possible size. This is advantageous if only the partition is required, for instance, for storing, adding, and multiplying the matrix by a vector. However, the cluster τ will usually not be a vertex in the cluster tree T_J. The latter property makes it difficult to define recursive block algorithms, which require a recursive subdivision of both the columns and the rows. If we want to apply algorithms that are defined on a hierarchy of block partitions, we have to use a partition that is constructed as the leaves of a so-called block cluster tree.

1.5 Block Cluster Trees

Each level of a cluster tree T_I contains a partition of the index set I. Since our aim is to find an admissible partition of $I \times J$, we consider cluster trees $T_{I\times J}$ for $I \times J$, which due to the type of their vertices will be referred to as **block cluster trees**.

Since the number of possible hierarchies of partitions is by far too large, we restrict ourselves to those partitions which are induced from subdividing the rows and columns. Let T_I and T_J be cluster trees for I and J with corresponding mappings S_I and S_J. We will consider block cluster trees $T_{I\times J}$ which are defined by the following mapping $S_{I\times J}$:

$$S_{I\times J}(t \times s) = \begin{cases} \varnothing, & \text{if } t \times s \text{ is admissible or } S_I(t) = \varnothing \text{ or } S_J(s) = \varnothing, \\ S_I(t) \times S_J(s), & \text{else.} \end{cases}$$

The depth $L(T_{I\times J})$ of the tree $T_{I\times J}$ is obviously bounded by the minimum of the depths of the generating cluster trees T_I and T_J; i.e.,

$$L(T_{I\times J}) \leq \min\{L(T_I), L(T_J)\}.$$

The leaves of $T_{I\times J}$ constitute an admissible partition $P := \mathscr{L}(T_{I\times J})$. Any sub-tree of $T_{I\times J}$ with root $t\times s \in T_{I\times J}$ is a block cluster tree (for $t\times s$) and will be denoted by $T_{t\times s}$. Its leaves $P' := \mathscr{L}(T_{t\times s})$ define an admissible partition of $t\times s$.

Usually, a block cluster is built from binary cluster trees; i.e., $\deg t = 2$ for $t \in T_I \setminus \mathscr{L}(T_I)$. In this case, $T_{I\times J}$ is a quad-tree due to the definition of $S_{I\times J}$. Note that each block $t\times s \in T_{I\times J}$ consists of clusters $t \in T_I$ and $s \in T_J$ from the same level in their respective cluster tree.

A measure for the complexity of a partition is the so-called sparsity constant; cf. [116]. The name of this constant stems from the analogy with sparse matrices, where the bounded number of nonzero entries per row or columns reduces the complexity. For hierarchical matrices the "sparsity" is the maximum number of blocks in the partition that are associated with a given row or column cluster.

Definition 1.35. Let T_I and T_J be cluster trees for the index sets I and J and let $T_{I\times J}$ be a block cluster tree for $I\times J$. We denote the number of blocks $t\times s \in T_{I\times J}$ associated with a given cluster $t \in T_I$ by

$$c^r_{\mathrm{sp}}(T_{I\times J}, t) := |\{s \subset J : t\times s \in T_{I\times J}\}|.$$

Similarly, for a given cluster $s \in T_J$

$$c^c_{\mathrm{sp}}(T_{I\times J}, s) := |\{t \subset I : t\times s \in T_{I\times J}\}|$$

stands for the number of blocks $t\times s \in T_{I\times J}$. The **sparsity constant** c_{sp} of a block cluster tree $T_{I\times J}$ is then defined as

$$c_{\mathrm{sp}}(T_{I\times J}) := \max\left\{\max_{t\in T_I} c^r_{\mathrm{sp}}(T_{I\times J}, t), \max_{s\in T_J} c^c_{\mathrm{sp}}(T_{I\times J}, s)\right\}.$$

Although the complexity of $\mathscr{H}$-matrices together with their arithmetic has been analyzed before the introduction of the sparsity constant (see [127, 133, 132, 17]), the usage of this constant is beneficial to the length and readability of the proofs. The following example shows that for elliptic problems c_{sp} is bounded independently of $|I|$. Note that for this result the sets X_i, $i \in I$, do not have to be quasi-uniform.

Example 1.36. Consider again Example 1.13 and assume that T_I and T_J are geometrically balanced, i.e., (1.22) is satisfied. Then $c_{\mathrm{sp}} \le 2\nu c_\Omega c_g c_G (2+1/\eta)^m$ holds.

Proof. Let $t \in T_I^{(\ell)}$ with an associated point $z_t \in X_t$. The estimates (1.20) and (1.22) guarantee that each neighborhood

$$N_\rho := \{s \in T_J^{(\ell)} : \max_{x\in X_s} |x - z_t| \le \rho\}, \quad \rho > 0,$$

of t contains at most $\nu c_G c_\Omega 2^\ell \rho^m$ clusters s from the same level ℓ in T_J. This follows from

$$|N_\rho| 2^{-\ell}/c_G \le \sum_{s\in N_\rho} \mu(X_s) \le \nu \mu(X_{N_\rho}) \le \nu c_\Omega \rho^m. \tag{1.32}$$

Let $s \in T_J$ such that $t \times s \in T_{I\times J}$. Furthermore, let t^* and s^* be the father clusters of t and s, respectively. Assume that $\max_{x\in X_s} |x - z_t| \geq \rho_0$, where

$$\rho_0 := \min\{\operatorname{diam} X_{t^*}, \operatorname{diam} X_{s^*}\}/\eta + \operatorname{diam} X_{t^*} + \operatorname{diam} X_{s^*} \leq (c_g 2^{-(\ell-1)})^{1/m}(2+1/\eta).$$

Then

$$\operatorname{dist}(X_{t^*}, X_{s^*}) \geq \max_{x\in X_s} |x - z_t| - \operatorname{diam} X_{t^*} - \operatorname{diam} X_{s^*} \geq \min\{\operatorname{diam} X_{t^*}, \operatorname{diam} X_{s^*}\}/\eta$$

implies that $t^* \times s^*$ is admissible. Thus, $T_{I\times J}$ cannot contain $t \times s$, which is a contradiction. It follows that

$$\max_{x\in X_s} |x - z_t| < \rho_0 \leq (c_g 2^{-(\ell-1)})^{1/m}(2+1/\eta).$$

From (1.32) we obtain that

$$c^r_{\mathrm{sp}} = |\{s \in T_J : t \times s \in P\}| \leq 2\nu c_\Omega c_g c_G (2+1/\eta)^m.$$

Interchanging the roles of t and s, one shows that c^c_{sp} is bounded by the same constant. □

Remark 1.37. Note that the boundedness of c_{sp} follows from (1.22). In particular, c_{sp} will be bounded for graded meshes. An overall linear complexity, however, can not be achieved for this kind of meshes since the depth of a geometrically balanced cluster tree will be of order n in general; cf. Example 1.28. There are special grids which are not quasi-uniform but which can still be shown to lead to almost linear complexity; cf. [131, 116].

The ideas of the previous proof can also be used to show that c_{sp} is bounded for partitions satisfying the algebraic admissibility condition (1.13); cf. [25]. Note that the power of this condition is that it does not involve the geometry of the discretization. Hence, the assumption that the cluster trees T_I and T_J are geometrically balanced can be omitted, which allows to treat general grids including adaptively refined ones. Instead, we can use cardinality balanced cluster trees. We assume that $I = J$ and that

$$|\mathcal{N}_\rho(t)| \leq c_V(\operatorname{diam} t + \rho)^m \quad \text{for all } \rho > 0, \tag{1.33}$$

where $\mathcal{N}_\rho(t) := \{i \in I : \operatorname{dist}(t,i) \leq \rho\}$. $\operatorname{diam} t$ and $\operatorname{dist}(t,s)$ were defined in (1.12). Note that condition (1.33) restricts the class of considered matrices to matrices which are in particular sparse.

Example 1.38. Under the assumptions (1.26), (1.27), and (1.33) it holds that $c_{\mathrm{sp}} \leq 2c_V c''_d C_b/c_b(2+1/\eta)^m$.

Proof. Let $t \in T_I^{(\ell)}$, $i_0 \in t$, and $N_\rho := \{s \in T_I^{(\ell)} : \max_{j\in s} d_{i_0 j} \leq \rho\}$ for $\rho > 0$. Using (1.26), we see from

$$|N_\rho| c_b |I| 2^{-\ell} \le \sum_{s \in N_\rho} |s| \le |\mathcal{N}_\rho(i_0)| \le c_V \rho^m \tag{1.34}$$

that the neighborhood N_ρ of t contains at most $c_V / c_b \rho^m 2^\ell / |I|$ clusters s from the same level ℓ in T_I.

Let $s \in T_I$ be such that $t \times s \in T_{I\times I}$. Furthermore, let t^* and s^* be the father clusters of t and s, respectively. Assume that $\max_{j\in s} d_{i_0 j} \ge \rho_0$, where

$$\rho_0 := \min\{\operatorname{diam} t^*, \operatorname{diam} s^*\}/\eta + \operatorname{diam} t^* + \operatorname{diam} s^* \le (c_d'' C_b |I| 2^{-(\ell-1)})^{1/m} (2 + 1/\eta).$$

Then

$$\operatorname{dist}(t^*, s^*) \ge \max_{j\in s} d_{i_0 j} - \operatorname{diam} t^* - \operatorname{diam} s^* \ge \min\{\operatorname{diam} t^*, \operatorname{diam} s^*\}/\eta$$

implies that $t^* \times s^*$ is admissible. Thus, $T_{I\times J}$ cannot contain $t \times s$, which is a contradiction. It follows that

$$\max_{j\in s} d_{i_0 j} < \rho_0 \le (c_d'' C_b |I| 2^{-(\ell-1)})^{1/m} (2 + 1/\eta).$$

From (1.34) we obtain that

$$c_{\mathrm{sp}}^r = |\{s \in T_I : t \times s \in P\}| \le 2 c_V c_d'' C_b / c_b (2 + 1/\eta)^m.$$

Again, by interchanging the roles of t and s, one shows that c_{sp}^c is bounded by the same constant. □

Many algorithms in the context of hierarchical matrices can be applied blockwise. In this case, the cost of the algorithm is the sum over the costs of each block. Assume that on each block the costs are bounded by a constant $c > 0$. Then

$$\sum_{t\times s \in T_{I\times J}} c \le c \sum_{t \in T_I} |\{s \subset J : t \times s \in T_{I\times J}\}| \le c\, c_{\mathrm{sp}}^r |T_I| \le c\, c_{\mathrm{sp}} |T_I|. \tag{1.35}$$

By interchanging the roles of t and s, we also obtain $\sum_{t\times s\in T_{I\times J}} c \le c\, c_{\mathrm{sp}} |T_J|$. If the cost of the algorithm on each block $t \times s \in P$ is bounded by $c(|t| + |s|)$, then the overall cost can be estimated as

$$\sum_{t\times s \in T_{I\times J}} c(|t| + |s|) = c \sum_{t \in T_I'} \sum_{\{s \in T_J : t\times s \in T_{I\times J}\}} |t| + c \sum_{s \in T_J'} \sum_{\{t \in T_I : t\times s \in T_{I\times J}\}} |s| \tag{1.36a}$$

$$\le c\, c_{\mathrm{sp}} \left(\sum_{t\in T_I'} |t| + \sum_{s \in T_J'} |s| \right) \le c\, c_{\mathrm{sp}} L(T_{I\times J}) [|I| + |J|] \tag{1.36b}$$

$$\le c\, c_{\mathrm{sp}} [L(T_I)|I| + L(T_J)|J|] \tag{1.36c}$$

due to Lemma 1.21. Here, T_I' and T_J' denote the sub-trees of T_I and T_J, respectively, which are actually used to construct $T_{I\times J}$; i.e.,

$$T_I' := \{t \in T_I : \text{there is } t' \subset t \text{ and } s' \in T_J \text{ such that } t' \times s' \in T_{I\times J}\}.$$

Lemma 1.39. *Let T_I and T_J be cluster trees for the index sets I and J, respectively. For the number of blocks in a partition $\mathscr{L}(T_{I\times J})$ and for the number of vertices in $T_{I\times J}$ it holds that*

$$|\mathscr{L}(T_{I\times J})| \leq |T_{I\times J}| \leq 2c_{\mathrm{sp}} \min\{|\mathscr{L}(T_I)|, |\mathscr{L}(T_J)|)\}.$$

If $|t| \geq n_{\min}$ for all $t \in T_I \cup T_J$, then $|\mathscr{L}(T_{I\times J})| \leq |T_{I\times J}| \leq 2c_{\mathrm{sp}} \min\{|I|, |J|\}/n_{\min}$.

Proof. Estimate (1.35) applied to $|T_{I\times J}| = \sum_{t\times s\in T_{I\times J}} 1$ gives

$$|T_{I\times J}| \leq c_{\mathrm{sp}} \min\{|T_I|, |T_J|\}.$$

Lemma 1.18 states that $|T_I| \leq 2|\mathscr{L}(T_I)|$ and $|T_J| \leq 2|\mathscr{L}(T_J)|$. □

Lemma 1.40. *Let T_I and T_J be balanced cluster trees. Taking into account requirement (ii) on the admissibility condition, the number of operations for constructing the block cluster tree is of the order $c_{\mathrm{sp}}[|I|\log|I| + |J|\log|J|]$.*

Proof. Use (1.36) and Lemma 1.20. □

The time required for computing an admissible matrix partition can be neglected compared with the rest of the computation. Using an admissible partition, we will define the set of hierarchical matrices in the following chapter.

Chapter 2
Hierarchical Matrices

In this chapter let T_I and T_J be binary cluster trees for the index sets I and J, respectively. A generalization to arbitrary cluster trees is possible; see Sect. 4.5 for nested dissection cluster trees, which are ternary. The block cluster tree $T_{I\times J}$ is assumed to be generated using a given admissibility condition as described in the previous chapter. We will define the set of hierarchical matrices originally introduced by Hackbusch [127] and Hackbusch and Khoromskij [133, 132]; see also [128]. The elements of this set can be stored with logarithmic-linear complexity and provide data-sparse representations of fully populated matrices. Additionally, combining the hierarchical partition and the efficient structure of low-rank matrices, an approximate algebra can be defined which is based on divide-and-conquer versions of the usual block operations. The efficient replacements for matrix addition and matrix multiplication can be used to define substitutes for higher level matrix operations such as inversion, *LU* factorization, and *QR* factorization.

After the definition of hierarchical matrices in Sect. 2.1, we consider the multiplication of such matrices by a vector in Sect. 2.2. It will be seen that this can be done with logarithmic-linear complexity. In Sect. 2.3 we describe how to perform this multiplication on a parallel computer. Matrix operations such as addition and multiplication and relations between local and global norm estimates were investigated in detail in [116]. We review the results and improve some of the estimates in Sect. 2.4, Sect. 2.5, and Sect. 2.7, since these results will be important for higher matrix operations. Furthermore, we adapt the proofs to our way of clustering and present an improved addition algorithm which can be shown to preserve positivity. An accelerated matrix multiplication can be defined (see Sect. 2.7.4) if one of the factors is semi-separable. In Sect. 2.6 we analyze a technique for reducing the computational costs of $\mathcal{H}$-matrix approximants by unifying neighboring blocks. In Sect. 2.8, Sect. 2.9, and Sect. 2.10 the $\mathcal{H}$-matrix inversion, *LU*, and *QR* factorization are presented. In Sect. 2.11 we point out the similarities of $\mathcal{H}^2$-matrices with fast multipole methods. Finally, in Sect. 2.12 we investigate the required accuracy of $\mathcal{H}$-matrix approximants if they are to be used for preconditioning. It will be seen that low-precision approximations will be sufficient to guarantee a bounded number of preconditioned iterations. $\mathcal{H}$-matrix preconditioners can be constructed in a

M. Bebendorf, *Hierarchical Matrices*. Lecture Notes in Computational Science and Engineering 63,

purely algebraic way using the hierarchical inverse or the hierarchical LU decomposition.

Since we do not want to exploit properties of the underlying operator at this point, the complexity is estimated in terms of the maximum rank among the blocks in the partition. The size of this rank will be analyzed depending on the prescribed accuracy in the second part of this book when more properties of the underlying operator can be accessed. If the blockwise rank is assumed to scale logarithmically with the number of degrees of freedom, all presented operations can be seen to have logarithmic-linear complexity.

2.1 The Set of Hierarchical Matrices

Definition 2.1. The set of **hierarchical matrices** on the block cluster tree $T_{I\times J}$ with admissible partition $P := \mathscr{L}(T_{I\times J})$ and blockwise rank k is defined as

$$\mathscr{H}(T_{I\times J},k) = \left\{A \in \mathbb{C}^{I\times J} : \operatorname{rank} A_b \leq k \text{ for all admissible } b \in P\right\}.$$

For the sake of brevity, elements from $\mathscr{H}(T_{I\times J},k)$ will often be called $\mathscr{H}$-matrices.

Remark 2.2. For an efficient treatment of admissible blocks the outer-product representation from Sect. 1.1.1 should be used. Additionally, it is advisable not to use the maximum rank k but the actual rank of the respective block as the number of rows and columns. Storing non-admissible blocks entrywise will increase the efficiency.

Example 2.3. The left picture of Fig. 2.1 shows an $\mathscr{H}$-matrix approximant for the matrix $a_{ij} = (|i-j|+1)^{-1}$, $i,j = 1,\ldots,n$. The right picture represents an approximant for the *Hilbert matrix* $h_{ij} = (i+j-1)^{-1}$, $i,j = 1,\ldots,n$. Depending on the kind of "singularity", the admissibility conditions from the Examples 1.13 and 1.12 have to be used. For the Hilbert matrix an approximant with accuracy $\varepsilon = 1_{10}-4$ can be found which for $n = 1\,000$ reduces the amount of storage to 3.2% and for $n = 100\,000$ to 0.32%.

A few easy consequences are gathered in the following two lemmas.

Lemma 2.4. *Let $A \in \mathscr{H}(T_{I\times J},k)$. Then*

(i) any submatrix A_b, $b \in T_{I\times J}$, belongs to $\mathscr{H}(T_b,k)$;
(ii) the transpose A^T and the Hermitian transpose A^H belong to $\mathscr{H}(T_{J\times I},k)$ provided that the admissibility condition is symmetric; i.e., any block $s\times t$ is admissible if $t\times s$ is admissible.

We define the set

$$\pi_I := \{t \in T_I : \exists s \in T_J \text{ such that } t\times s \in P \text{ and } \forall t' \subset t \text{ and } \forall s' \in T_J : t'\times s' \notin P\},$$

which is the finest partition of I made from clusters appearing in the partition P.

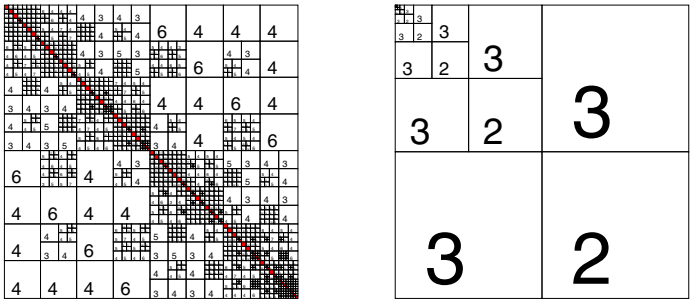


Fig. 2.1 $\mathscr{H}$-matrices with their blockwise ranks.

Lemma 2.5. *Let D_1, D_2 be block diagonal matrices on the tensor partitions $\pi_I \times \pi_I$ and $\pi_J \times \pi_J$, respectively. Then $D_1 A D_2 \in \mathscr{H}(T_{I\times J}, k)$ provided that $A \in \mathscr{H}(T_{I\times J}, k)$.*

Proof. Due to the assumption, each block $t \times s \in P$ of $D_1 A D_2$ arises from multiplying A_{ts} by $(D_1)_{tt}$ and $(D_2)_{ss}$. Hence, from Theorem 1.1 we have that $\operatorname{rank}(D_1 A D_2)_{ts} = \operatorname{rank}(D_1)_{tt} A_{ts} (D_2)_{ss} \le \operatorname{rank} A_{ts}$. □

We have already seen in Sect. 1.1.1 that the storage requirements for an admissible block $b = t \times s \in \mathscr{L}(T_{I\times J})$ of $A \in \mathscr{H}(T_{I\times J}, k)$ are

$$N_{\mathrm{st}}(A_b) \le k(|t| + |s|).$$

A non-admissible block A_b, $b \in P$, is stored entrywise and thus requires $|t||s|$ units of storage. Since $\min\{|t|,|s|\} \le n_{\min}$ (see Remark 1.17) we have

$$|t||s| = \min\{|t|,|s|\} \max\{|t|,|s|\} \le n_{\min}(|t| + |s|). \tag{2.1}$$

Hence, for storing A_b, $b \in P$, at most $\max\{k, n_{\min}\}(|t| + |s|)$ units of storage are required. Using (1.36), we obtain the following theorem.

Theorem 2.6. *Let c_{sp} be the sparsity constant for the tree $T_{I\times J}$; cf. Definition 1.35. The storage requirements N_{st} for $A \in \mathscr{H}(T_{I\times J}, k)$ are bounded by*

$$N_{\mathrm{st}}(A) \le c_{\mathrm{sp}} \max\{k, n_{\min}\}[L(T_I)|I| + L(T_J)|J|].$$

If T_I and T_J are balanced cluster trees (cf. Definition 1.19), we have

$$N_{\mathrm{st}}(A) \sim \max\{k, n_{\min}\}[|I| \log |I| + |J| \log |J|].$$

Remark 2.7. Although $\mathscr{H}$-matrices are primarily aiming at dense matrices, sparse matrices A which vanish on admissible blocks are also in $\mathscr{H}(T_{I\times J}, n_{\min})$. Since the size of one of the clusters corresponding to non-admissible blocks is less than or equal to $n_{\min}$, the rank of each block A_b does not exceed $n_{\min}$. A deeper analysis

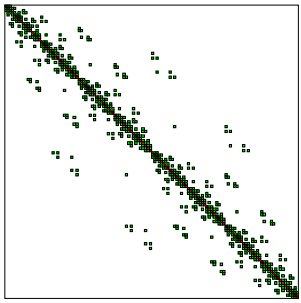


Fig. 2.2 A sparse $\mathscr{H}$-matrix (nonzero blocks are shown).

(see Lemma 4.2) shows that finite element discretizations can actually be stored with linear complexity.

2.2 Matrix-Vector Multiplication

The cost of multiplying an $\mathscr{H}$-matrix $A \in \mathscr{H}(T_{I\times J},k)$ or its Hermitian transpose A^H by a vector x is inherited from the blockwise matrix-vector multiplication:

$$Ax = \sum_{t\times s\in P} A_{ts}x_s \quad \text{and} \quad A^H x = \sum_{t\times s\in P} (A_{ts})^H x_t. \tag{2.2}$$

Since each admissible block $t\times s$ has the outer product representation $A_{ts} = UV^H$, $U \in \mathbb{C}^{t\times k}$, $V \in \mathbb{C}^{s\times k}$, at most $2k(|t|+|s|)$ operations are required to compute the matrix-vector products $A_{ts}x_s = UV^H x_s$ and $(A_{ts})^H x_t = VU^H x_t$. If $t\times s$ is non-admissible, then A_{ts} is stored entrywise and $\min\{|t|,|s|\} \le n_{\min}$. As in (2.1) we see that in this case

$$2|t||s| \le 2n_{\min}(|t|+|s|)$$

arithmetic operations are required. The same arguments that were used for Theorem 2.6 give the following theorem.

Theorem 2.8. *For the number of operations N_{MV} required for one matrix-vector multiplication Ax of $A \in \mathscr{H}(T_{I\times J},k)$ by a vector $x \in \mathbb{C}^J$ it holds that*

$$N_{\mathrm{MV}}(A) \le 2c_{\mathrm{sp}} \max\{k,n_{\min}\}[L(T_I)|I| + L(T_J)|J|].$$

If T_I and T_J are balanced cluster trees, we have

$$N_{\mathrm{MV}}(A) \sim \max\{k,n_{\min}\}[|I|\log|I| + |J|\log|J|].$$

Hence, $\mathscr{H}$-matrices are well suited for iterative schemes such as Krylov subspace methods (see [221]), which the matrix enters only through the matrix-vector product.

Obviously, the matrix-vector multiplication can also be done by a recursion through the block cluster tree $T_{I\times J}$. Since arithmetic operations are performed only on the leaves of $T_{I\times J}$, summing over the leaves of $T_{I\times J}$ as it is done in (2.2) is slightly more efficient. The latter representation is also more convenient for the following parallelization of the matrix-vector multiplication.

2.3 Parallel Matrix-Vector Multiplication

Although the product Ax of a matrix $A \in \mathscr{H}(T_{I\times J}, k)$ and a vector $x \in \mathbb{C}^J$ can be done with almost linear complexity, it can still be helpful to further reduce the execution time especially if many matrix-vector products are to be computed as part of an iterative solver, for instance. The parallelization of algorithms is always a promising way to achieve this reduction provided that significant parts of the algorithm admit independent execution. The following ideas were presented in [29].

Instead of a pure matrix-vector multiplication, in this section we examine the more general update of a vector $y \in \mathbb{C}^I$ by the operation

$$y := \alpha Ax + \beta y$$

with scalars $\alpha, \beta \in \mathbb{C}$. For this purpose we present two algorithms, one for distributed and the other for shared memory systems. For their description we use a unified model of a general parallel computer, a so-called **bulk synchronous parallel (BSP) computer**; see [253]. This model of a parallel system is based on three parameters: the number of processors p, the number of time steps for a global synchronization ℓ, and the ratio g of the total number of operations performed on all processors and the total number of words delivered by the communication network per second. All parameters are normalized with respect to the number of time steps per second.

A BSP computation consists of single *supersteps*. Each superstep has an *input phase*, a *local computation phase* and an *output phase*; see Fig. 2.3. During the input phase, each processor receives data sent during the output phase of the previous superstep. While all processors are synchronized after each superstep, all computations within each superstep are asynchronous.

The complexity of a BSP computation can be described by the parameters ℓ, g, the number of operations done on each processor, and the amount of data sent between the processors. The amount of work for a single superstep can be expressed by $w + (h_{\text{in}} + h_{\text{out}}) \cdot g + \ell$, where w is the maximum number of operations performed and h_{in}, h_{out} are the maximum numbers of data units received and sent by each processor, respectively. The total work of the BSP computation is the sum of the costs

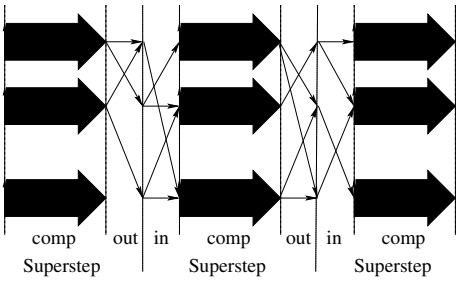


Fig. 2.3 BSP computation.

in each superstep, yielding an expression of the form $W + H \cdot g + S \cdot \ell$, where S is the number of supersteps.

The complexity analysis of the presented methods will be done in terms of parallel speedup and parallel efficiency.

Definition 2.9. Let $t(p)$ denote the time needed by a parallel algorithm with p processors. Then

$$S(p) := \frac{t(1)}{t(p)}$$

denotes the **parallel speedup** and

$$E(p) := \frac{S(p)}{p} = \frac{t(1)}{p \cdot t(p)}$$

its **parallel efficiency**.

We assume that both vectors x and y are distributed uniformly among p processors, each holding $|J|/p$ and $|I|/p$ entries of x and y, respectively. By x_q and y_q we denote the part of x and y on processor $0 \leq q < p$. This distribution ensures optimal complexity of all vector operations.

2.3.1 Parallelization for Usual Matrices

As a first step towards the hierarchical matrix-vector multiplication on a parallel machine we review the ideas of the BSP algorithm for dense matrices described in [181].

Consider the case of a dense matrix $A \in \mathbb{C}^{I \times J}$. Let each of the p processors hold a block A_q of size $(|I|/\sqrt{p}) \times (|J|/\sqrt{p})$ from a uniform partition of $I \times J$. The BSP algorithm is split into three steps. In the first step, each processor has to receive $|J|/\sqrt{p}$ entries of x needed for the local matrix-vector multiplication, which is done in the second superstep. The resulting entries of y are afterwards sent to the corresponding

processors such that in the third step all local coefficients of y can be summed up for the final result.

```
procedure dense_mult(α, A_q, x, β, y, q)
    { first step }
    y_q := β · y_q;
    send x_q to all processors sharing it;
    sync();
    { second step }
    y'_q := αA_q x_q;
    send respective parts of y'_q to all processors sharing it;
    sync();
    { third step }
    Y_q := {received vectors of local results};
    y_q := y_q + Σ_{y'_j ∈ Y_q} y'_j;
    sync();
end;
```

Algorithm 2.1: Dense matrix-vector multiplication.

The costs of Algorithm 2.1 are $|I|/p + g \cdot |J|/\sqrt{p} + \ell$ for scaling y and sending x in the first step, $|I||J|/p + g \cdot |I|/\sqrt{p} + \ell$ for the local matrix-vector product in the second step, and $|I|/\sqrt{p} + \ell$ for the summation in the last superstep. Therefore, the total costs required to multiply a dense matrix by a vector can be estimated as

$$\mathcal{O}\left(\frac{|I||J|}{p} + \frac{|I|+|J|}{\sqrt{p}}\right) + g \cdot \mathcal{O}\left(\frac{|I|+|J|}{\sqrt{p}}\right) + 3 \cdot \ell, \tag{2.3}$$

which can be shown to be optimal with respect to computation and communication costs; cf. [181].

2.3.2 Non-Uniform Block Distributions

The uniform block distribution which was used in the previous section is not suitable for $\mathcal{H}$-matrices since the costs of a matrix-vector multiplication vary among the blocks of an $\mathcal{H}$-matrix due to their different sizes and their different representations. Unfortunately, changing the distribution pattern is likely to result in communication costs which are no longer optimal. For instance, the random distribution in Fig. 2.4 (left) results from applying **list scheduling**, i.e., assigning the next not yet executed job to the first idle processor. Although list scheduling guarantees an efficient local multiplication phase in the second step of Algorithm 2.1, the vector x has to be sent to all processors with communication costs of $\mathcal{O}(|J|)$ due to the scattering of the matrix blocks across the whole $\mathcal{H}$-matrix. Furthermore, p vectors have to be summed up in the third step with computational costs $\mathcal{O}(|I|)$. Such a situation should therefore be avoided.

In order to be able to measure the communication and computation costs with respect to the vectors x and y, we introduce the *sharing constant* of of block

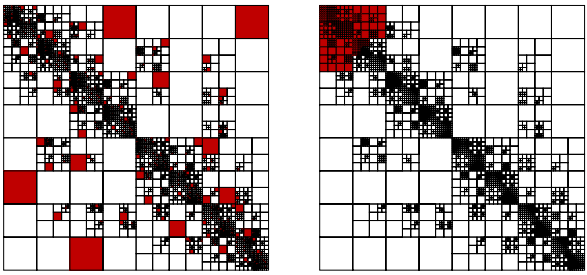


Fig. 2.4 List scheduling (left) versus sequence partitioning (right).

distribution, i.e., the maximum number of processors sharing one row or one column of A.

Definition 2.10. Let P be a partition of $I \times J$ and let P_q denote the blocks in P assigned to processor q. We define the **sharing constant** c_{sh} of P as

$$c_{\text{sh}} = \max_{i \in I,\, j \in J} \{c^r_{\text{sh}}(i), c^c_{\text{sh}}(j)\}, \tag{2.4}$$

where $c^r_{\text{sh}}(i) := |\{q \,|\, \exists t \times s \in P_q : i \in t\}|$ for $i \in I$ and $c^c_{\text{sh}}(j) := |\{q \,|\, \exists t \times s \in P_q : j \in s\}|$ for $j \in J$.

The constant c_{sh} can be used to express the costs for sending x in the first step and for summing up all local vectors y_q in the last step of Algorithm 2.1. The following definition allows to describe the costs for receiving the vector x and sending the local result y_q.

Definition 2.11. Let P be a partition of $I \times J$ and let P_q denote the blocks in P assigned to processor q. We define

$$I(q) = \bigcup_{t \times s \in P_q} t \quad \text{and} \quad J(q) = \bigcup_{t \times s \in P_q} s$$

as the rows and the columns associated with processor q. Furthermore, let

$$\rho = \max_{0 \le q < p} \{|I(q)|, |J(q)|\}.$$

For the above uniform block distribution c_{sh} equals $\sqrt{p}$, whereas the random distribution induced by list scheduling results in a constant c_{sh} which is of order p. Similarly, we find $\rho = \max\{|I|, |J|\}/\sqrt{p}$ in the case of a uniform partition and $\rho = \mathcal{O}(\max\{|I|, |J|\})$ for the random distribution resulting from list scheduling. Using c_{sh} and ρ, (2.3) can be rewritten to obtain the following complexity of the matrix-vector multiplication for general block distributions

$$\mathcal{O}\left(\frac{N_{\text{MV}}(A)}{p} + c_{\text{sh}}\frac{|I|+|J|}{p}\right) + g \cdot \mathcal{O}\left(c_{\text{sh}}\frac{|I|+|J|}{p} + \rho\right) + 3 \cdot \ell. \tag{2.5}$$

Due to the definition of $\mathscr{H}$-matrices, i.e., due to the admissibility condition, large blocks tend to be of the order $I \times J$. Although exactly this leads to efficient algorithms, it also restricts the possibility of reducing ρ. Hence, without splitting large matrix blocks, one always ends up with $\rho = \mathscr{O}(\max\{|I|, |J|\})$.

Fortunately, this negative result does not apply to c_{sh}, which can be reduced using **space-filling curves** and **sequence partitioning**. These methods produce a distribution of P with a much higher locality of the blocks associated to a specific processor q; see Fig. 2.4 (right). Due to the "compactness" of the sets P_q, the frequency of sharing an index with another processor is reduced.

2.3.2.1 Load Balancing with Sequence Partitioning

In this section it will be described how to distribute the blocks among the processors such that on one hand the numerical work for the processors are almost equal and on the other hand c_{sh} and ρ are small. In order to be able to balance the work, we first have to know the costs associated with a processor. Depending on the representations of dense and low-rank matrix blocks, the costs of each block are

$$c_{\text{MV},k}(t,s) = \begin{cases} |t| \cdot |s|, & \text{if } t \times s \in P \text{ is non-admissible,} \\ k(|t| + |s|), & \text{if } t \times s \in P \text{ is admissible.} \end{cases} \tag{2.6}$$

Assume that the set of blocks P has been rearranged as a sequence. A block distribution will be generated by subdividing this sequence into p pieces of comparable costs.

Definition 2.12. Let $C = \{c_1, c_2, \ldots, c_n\}$ be a sequence of costs $c_i > 0$. Furthermore, let $R = \{r_0, \ldots, r_p\}$ with $1 = r_0 \le r_1 \le \ldots \le r_p = n+1$, $r_i \in \mathbb{N}$, $0 < i < p$. Then R is called a **sequence partition** of (C, p). R is **optimal** with respect to (C, p) if for all partitions $R' = \{r'_0, \ldots, r'_p\}$ of C it holds that

$$\max_{0 \le i < p} \sum_{j=r'_i}^{r'_{i+1}-1} c_j \ge \max_{0 \le i < p} \sum_{j=r_i}^{r_{i+1}-1} c_j =: c_{\max}(C).$$

For the computation of an optimal partition of a sequence C, the knowledge of $c_{\max}(C)$, the costs of the most expensive interval in an optimal partition, is sufficient. In [195] an algorithm is presented which computes $c_{\max}(C)$ with complexity $\mathscr{O}(n \cdot p)$. An optimal partition can then be obtained by summing up the costs of each element of the list and starting a new subsequence whenever the costs exceed $c_{\max}(C)$.

The required sequence of the blocks in P can be generated using space filling curves. These curves describe a surjective mapping from the unit interval $[0,1]$ to the unit square $[0,1]^2$. Two examples of such curves, the **Z-** and the **Hilbert-curve**, are presented in Fig. 2.5. Since partitions of $I \times J$ can be mapped to the unit square, the order in which a leaf is reached by the curve defines a sequence usable for sequence partitioning. The neighborhood relationship of adjacent subintervals

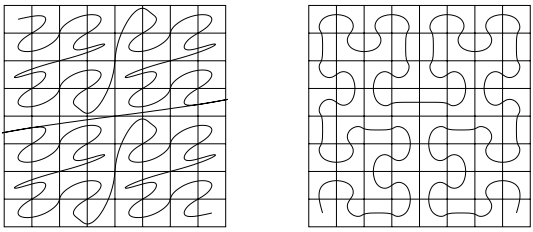


Fig. 2.5 Space-filling curves.

of space-filling curves guarantees the "compactness" of the corresponding sets P_q. The application of the Z- and the Hilbert-curve to a block partition is depicted in Fig. 2.6.

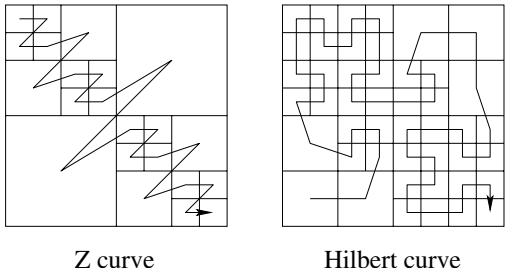


Fig. 2.6 Space-filling curves applied to $\mathscr{H}$-matrices.

The restriction to quadtrees in the definition of block cluster trees allows a simple computation of the ordering induced by space-filling curves. The basic algorithm is a depth first search (DFS) (see [249]) in $T_{I\times J}$. In contrast to the usual DFS algorithm, the order in which the sons $S(b)$ of a node $b \in T_{I\times J}$ are accessed is defined by a *mark* associated with each node. The marks and the corresponding order of the sons for the Z- and the Hilbert-curve is presented in Fig. 2.7. Here, the root of the block cluster tree always has the mark "A".

The motivation of load balancing with sequence partitioning was the reduction of the sharing constant c_{sh} compared with a random distribution generated by list scheduling. The value of c_{sh} obtained using the Z- and the Hilbert-curve for different numbers of processors is shown in Fig. 2.8 (left). For both space-filling curves one can observe a behavior of the kind $c_{\mathrm{sh}} \sim \sqrt{p}$, which is equal to the uniform distribution in the case of dense matrices. This shows the reduction of c_{sh} in comparison to a random distribution.

We compare the proposed distribution of blocks with another standard scheduling method which is not based on space-filling curves. Instead of assigning the blocks

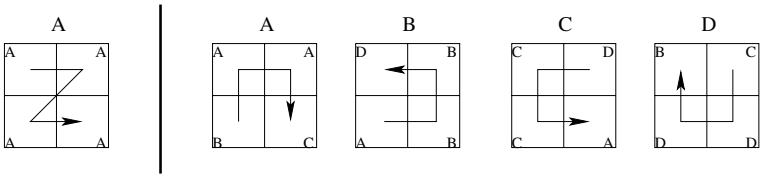


Fig. 2.7 Construction of space-filling curves: Z (left) and Hilbert (right).

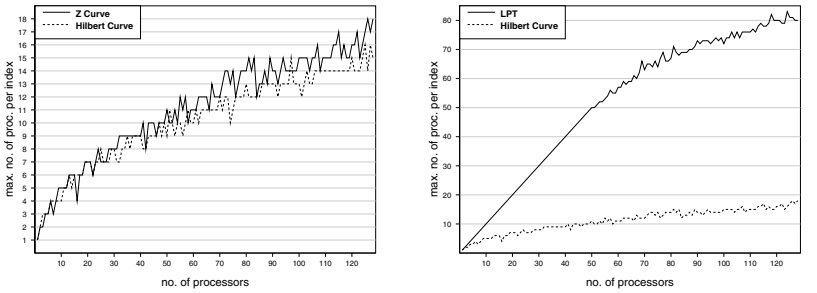


Fig. 2.8 Value of c_{sh} for space-filling curves and LPT scheduling.

randomly to the processors, **longest process time (LPT) scheduling** (cf. [113]) orders the blocks according to their costs, which usually results in a better load balancing than list scheduling. However, it does not reduce c_{sh}; see Fig. 2.8 (right). An $\mathcal{O}(p)$ dependence of c_{sh} is visible especially for small p. The number of blocks per processor becomes smaller if $p > 50$. Therefore, less processors share the same index.

2.3.2.2 Shared Memory Systems

Although communication costs can be neglected on shared memory systems, i.e., it can safely be assumed that $\ell = g = 0$, we can use the same algorithm as in the case of a distributed memory machine. This can be justified by examining the (hidden) constants in the part of (2.5) which describes the computational work. Assuming $c_{\text{sh}} \sim \sqrt{p}$, we can rewrite this equation as

$$\mathcal{O}\left(\frac{N_{\text{MV}}(A)}{p} + c\,\frac{|I|+|J|}{\sqrt{p}}\right)$$

with a small constant $c > 0$. On shared memory systems usual values for p range from 1 to 128. Since the influence of the second term can be seen only for large p, the first term dominates the computational work. Hence, a high parallel efficiency can also be expected on shared memory systems.

Algorithm 2.1 can be simplified using a threadpool (cf. [165]) based on POSIX threads. For this, the first two steps of the BSP algorithm, i.e., scaling y and the matrix-vector multiplication, are combined because the vector x can be accessed by all processors. The summation of the final result is done in a second step after all threads have computed the corresponding local results y_i'. Another advantage of this algorithm is that for the implementation only minor modifications of an existing sequential version are necessary, e.g., the computation of the matrix-vector product in the first step differs only by the involved set of matrix blocks.

```
procedure step_1(q, β, y_q, A_q, x)
    y_q := β · y_q;
    y'_q := α A_q x;
end;

procedure step_2(q, y_q)
    Y_q := {y'_j | I(j) ∩ I(q) ≠ ∅};
    y_q := y_q + Σ_{y'_j ∈ Y_q} y'_j|_{I(q)};
end;

procedure tp_mv_mul(α, A, x, β, y)
    for 0 ≤ q < p do run( step_1(q, β, y_q, A_q, x) );
    sync_all();
    for 0 ≤ q < p do run( step_2(q, y_q) );
    sync_all();
end;
```

Algorithm 2.2: Matrix-vector multiplication using threads.

2.3.3 Numerical Experiments

In this section we examine the performance of the presented parallel matrix-vector multiplication. For simplicity the factors α and β in the product $y := \alpha Ax + \beta y$ are chosen 1. In all examples the time for 100 matrix-vector multiplications was measured. We apply the proposed methods to $\mathscr{H}$-matrices stemming from the Galerkin discretization of the integral operator from Example 3.43.

Remark 2.13. Parallelizing the matrix-vector product cannot be regarded independently of other operations such as generating the matrix approximant. Computing $\mathscr{H}$-matrix approximations is usually much more time-consuming than multiplying the approximant by a vector. Therefore, an algorithm for the approximation of discrete integral operators together with its parallelization is presented in Sect. 3.4. Since we should not reassign the blocks to the processors after they have been generated, we will use the block distribution of the matrix-vector multiplication when approximating the matrix. Note that the blocks can be approximated independently while for the matrix-vector multiplication the "compactness" is a critical issue. Here, the problem arises that the rank k in (2.6) is not known before the matrix

has been generated if a required accuracy has to be satisfied. In this case, we replace k in (2.6) by a constant $k_{\text{avg}} = 10$.

2.3.3.1 Shared Memory Systems

For the experiments on a shared memory system an HP9000, PA-RISC with 875 MHz was used. The first comparisons were done employing a square $\mathscr{H}$-matrix approximant having a fixed rank of $k = 10$ on each admissible block. The CPU times resulting from using Algorithm 2.2 and the corresponding parallel efficiency are presented in Table 2.1. The weak parallel performance for small problem sizes $|I|$

Table 2.1 100 parallel matrix-vector multiplications for fixed $k = 10$.

	$p=1$	$p=4$		$p=8$		$p=12$		$p=16$	
$\|I\|$	time	time	E	time	E	time	E	time	E
3 968	11.7s	3.3s	88%	1.8s	80%	1.3s	73%	1.3s	57%
7 920	30.9s	8.5s	91%	4.6s	84%	3.5s	74%	2.9s	66%
19 320	94.9s	25.9s	92%	13.5s	88%	9.5s	83%	7.5s	79%
43 680	251.7s	70.9s	89%	36.0s	87%	23.9s	88%	18.9s	84%
89 400	556.4s	152.2s	91%	80.0s	87%	53.4s	87%	41.1s	85%
184 040	1277.5s	347.7s	92%	186.0s	86%	120.1s	89%	97.5s	82%

is probably due to the sequential parts in the algorithm, i.e., management overhead. Since this part remains constant independently of the problem size, the parallel efficiency grows with $|I|$ and stabilizes at about 80–90 %.

Table 2.2 shows the results for the same operation but with an $\mathscr{H}$-matrix obtained from approximation with fixed accuracy $\varepsilon = 1_{10}{-}4$ but variable rank k. The same

Table 2.2 100 parallel matrix-vector multiplications for variable k.

	$p=1$	$p=4$		$p=8$		$p=12$		$p=16$	
$\|I\|$	time	time	E	time	E	time	E	time	E
3 968	9.6s	2.6s	93%	1.6s	73%	1.3s	61%	1.3s	48%
7 920	23.8s	6.3s	95%	3.7s	80%	3.1s	65%	2.4s	63%
19 320	66.7s	17.2s	97%	9.6s	87%	7.0s	80%	5.6s	74%
43 680	169.6s	44.6s	95%	22.8s	93%	15.7s	90%	13.0s	82%
89 400	346.2s	91.1s	95%	47.1s	92%	32.8s	88%	25.7s	84%
184 040	780.5s	202.6s	96%	107.1s	91%	69.8s	93%	55.0s	89%

behavior as in the previous table is visible: the parallel efficiency grows with $|I|$ and reaches an almost optimal value of about 90 %.

2.3.3.2 Distributed Memory Systems

The following tests were carried out on an AMD Athlon 900 MHz cluster. Due to memory restrictions, problems for large $|I|$ could not be computed with a small number of processors p. The corresponding parallel efficiency for these problem sizes is therefore computed with respect to the smallest available p, i.e.,

$$E(p) := \frac{p' \cdot t(p')}{p \cdot t(p)},$$

where p' denotes the smallest number of processors which was able to compute the problem. The presented storage size in these cases is approximated by $p' N_{\mathrm{st}}$, where N_{st} denotes the memory consumption per processor on a system with p' CPUs. The results from Table 2.3 were obtained for fixed rank $k = 10$. One observes the same

Table 2.3 100 parallel matrix-vector multiplications for fixed $k = 10$.

	$p=1$	$p=4$		$p=8$		$p=12$		$p=16$	
$\|I\|$	time	time	E	time	E	time	E	time	E
3 968	16.2s	4.8s	85%	2.8s	72%	2.1s	65%	1.6s	65%
7 920	43.8s	12.1s	91%	6.7s	81%	4.8s	76%	4.0s	69%
19 320	141.1s	39.5s	89%	20.2s	87%	14.7s	80%	11.1s	80%
43 680		107.9s		57.0s	*95%*	42.0s	*86%*	32.1s	*84%*
89 400				129.9s		90.8s	*95%*	69.7s	*93%*
184 040						209.4s		157.4s	*100%*

behavior for the parallel performance as in the case of a shared memory system: a better efficiency is obtained for larger $|I|$. This effect is also visible for fixed accuracy $\varepsilon = 1_{10}{-}4$ as the results in Table 2.4 indicate. Due to the approximation of the actual

Table 2.4 100 parallel matrix-vector multiplications for variable k.

	$p=1$	$p=4$		$p=8$		$p=12$		$p=16$	
$\|I\|$	time	time	E	time	E	time	E	time	E
3 968	12.0s	3.6s	83%	2.1s	71%	2.0s	50%	1.3s	57%
7 920	30.0s	8.6s	87%	5.0s	76%	3.6s	69%	3.0s	64%
19 320	84.6s	27.0s	79%	13.2s	80%	9.5s	74%	7.5s	70%
43 680	221.0s	64.0s	86%	34.5s	80%	23.2s	80%	25.5s	54%
89 400				74.1s		53.6s	*92%*	42.4s	*87%*
184 040						119.6s		90.8s	*99%*

costs (see Remark 2.13), the parallel efficiency is not as high as for an $\mathscr{H}$-matrix with fixed rank.

As a conclusion of these test, we observe that starting from an existing sequential implementation of $\mathscr{H}$-matrices, only a minimal programming effort is necessary to

make use of multiple processors on a shared memory machine. The resulting algorithms show a high parallel efficiency and are therefore recommended for the acceleration of the $\mathscr{H}$-matrix arithmetic on workstations and compute-servers. If a larger number of processors is needed, distributed memory machines are usually preferred due to their lower costs. Using the BSP model, the design and implementation of parallel algorithms on such computer systems is similar to shared memory systems. The corresponding parallel matrix-vector multiplication also shows a high parallel efficiency if the problem size is sufficiently large.

While the matrix-vector multiplication is exact up to machine precision, the following replacements of the usual matrix operations are approximate. Since most of the algorithms guarantee a prescribed accuracy on each block, it is important to be able to relate blockwise accuracy estimates to global ones.

2.4 Blockwise and Global Norms

From the analysis we will usually obtain estimates on each of the blocks b of a partition P. However, such estimates are finally required for the whole matrix. If we are interested in the Frobenius norm, blockwise estimates directly translate to global estimates:

$$\|A_b\|_F \leq \varepsilon \text{ for all } b \in P \quad \Longrightarrow \quad \|A\|_F \leq \sqrt{|P|}\,\varepsilon$$

and

$$\|A_b\|_F \leq \|B_b\|_F \text{ for all } b \in P \quad \Longrightarrow \quad \|A\|_F \leq \|B\|_F.$$

Both implications follow from

$$\|A\|_F^2 = \sum_{b \in P} \|A_b\|_F^2. \tag{2.7}$$

For the spectral norm the situation is a bit more difficult. We can, however, exploit the structure of the partition P together with the following lemma.

Lemma 2.14. *Consider the following $r \times r$ block matrix*

$$A = \begin{bmatrix} A_{11} & \dots & A_{1r} \\ \vdots & & \vdots \\ A_{r1} & \dots & A_{rr} \end{bmatrix} \tag{2.8}$$

with $A_{ij} \in \mathbb{C}^{m_i \times n_j}$, $i, j = 1, \dots, r$. Then it holds that

$$\max_{i,j=1,\dots,r} \|A_{ij}\|_2 \leq \|A\|_2 \leq \left(\max_{i=1,\dots,r} \sum_{j=1}^{r} \|A_{ij}\|_2 \right)^{1/2} \left(\max_{j=1,\dots,r} \sum_{i=1}^{r} \|A_{ij}\|_2 \right)^{1/2}. \tag{2.9}$$

Proof. Let $x = [x_1, \dots, x_r]^T \in \mathbb{C}^n$, where $x_j \in \mathbb{C}^{n_j}$, $j = 1, \dots, r$, and $n := \sum_{j=1}^{r} n_j$. Observe that

$$\|Ax\|_2^2 = \sum_{i=1}^{r} \| \sum_{j=1}^{r} A_{ij} x_j \|_2^2 \leq \sum_{i=1}^{r} \left(\sum_{j=1}^{r} \|A_{ij}\|_2 \|x_j\|_2 \right)^2 = \|\hat{A}\hat{x}\|_2^2,$$

where $\hat{A} \in \mathbb{R}^{r \times r}$ has the entries $\hat{a}_{ij} = \|A_{ij}\|_2$ and $\hat{x} \in \mathbb{R}^r$ is the vector with components $\hat{x}_j = \|x_j\|_2$, $j = 1, \ldots, r$. It is well known that $\|\hat{A}\|_2^2 \leq \|\hat{A}\|_1 \|\hat{A}\|_\infty$. Hence,

$$\|\hat{A}\hat{x}\|_2^2 \leq \|\hat{A}\|_1 \|\hat{A}\|_\infty \|\hat{x}\|_2^2 = \|\hat{A}\|_1 \|\hat{A}\|_\infty \|x\|_2^2$$

gives the first part of the assertion. The lower bound follows from the fact that the spectral norm of any sub-block of a matrix A is bounded by the spectral norm of A. □

For block matrices generated from recursive subdivision we obtain

Theorem 2.15. *Assume that the partition P is generated from $I \times J$ by recursively subdividing each block into a 2×2 block structure at most L times. Furthermore, let $A, B \in \mathbb{C}^{I \times J}$ such that $\|A_b\|_2 \leq \|B_b\|_2$ for all blocks $b \in P$. Then it holds that*

$$\|A\|_2 \leq 2^L \|B\|_2.$$

Proof. The assertion is proved by induction over the depth L of the cluster tree. The estimate is trivial if $L = 0$. Assume that the assertion holds for an $L \in \mathbb{N}$. Let

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

have depth $L+1$. Since A_{ij}, $i, j = 1, 2$, have depth L, we know from the induction that

$$\|A_{ij}\|_2 \leq 2^L \|B_{ij}\|_2, \quad i, j = 1, 2.$$

The previous lemma shows

$$\|A\|_2 = \| \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \|_2 \leq 2 \max_{i,j=1,2} \|A_{ij}\|_2 \leq 2^{L+1} \max_{i,j=1,2} \|B_{ij}\|_2 \leq 2^{L+1} \|B\|_2,$$

which proves the assertion. □

For the usual choice $L \sim \min\{\log_2 |I|, \log_2 |J|\}$, the coefficient 2^L in the previous theorem will be of the order $\min\{|I|, |J|\}$. If more structure of P than just a recursive subdivision is known, then this estimate can be significantly improved. Note that block cluster trees have $|T_{I \times J}| \sim \min\{|I|, |J|\}$ elements while recursive subdivision in general leads to $|I| \cdot |J|$ blocks.

An important consequence of (2.9) is that for matrices (2.8) vanishing in all but ν blocks in each row and each column it follows that

$$\max_{i,j=1,\ldots,r} \|A_{ij}\|_2 \leq \|A\|_2 \leq \nu \max_{i,j=1,\ldots,r} \|A_{ij}\|_2.$$

The previous estimate was also proved in [114] with a different technique. This equivalence of the global and the blockwise spectral norm is useful in translating blockwise errors to a global one. When relative error estimates are to be derived, we will additionally need to estimate how a local norm relation is carried over to the whole matrix.

Theorem 2.16. *Let P be the leaves of a block cluster tree $T_{I\times J}$. Then for $A,B \in \mathscr{H}(T_{I\times J},k)$ it holds that*

(i) $\max_{b\in P}\|A_b\|_2 \le \|A\|_2 \le c_{\mathrm{sp}} L(T_{I\times J}) \max_{b\in P}\|A_b\|_2$;
(ii) $\|A\|_2 \le c_{\mathrm{sp}} L(T_{I\times J})\|B\|_2$ *provided* $\max_{b\in P}\|A_b\|_2 \le \max_{b\in P}\|B_b\|_2$.

Proof. Let A_ℓ denote the part of A which corresponds to the blocks of P from the ℓth level of $T_{I\times J}$; i.e.,

$$(A_\ell)_b = \begin{cases} A_b, & b \in T_{I\times J}^{(\ell)} \cap P, \\ 0, & \text{else.} \end{cases}$$

Then $A = \sum_{\ell=1}^{L(T_{I\times J})} A_{\ell-1}$. Since A_ℓ has tensor structure with at most c_{sp} blocks per block row or block column, Lemma 2.14 gives $\|A_\ell\|_2 \le c_{\mathrm{sp}} \max_{b\in T_{I\times J}^{(\ell)}\cap P}\|A_b\|_2$, such that

$$\|A\|_2 \le \sum_{\ell=1}^{L(T_{I\times J})} \|A_{\ell-1}\|_2 \le c_{\mathrm{sp}} \sum_{\ell=1}^{L(T_{I\times J})} \max_{b\in T_{I\times J}^{(\ell-1)}\cap P} \|A_b\| \le c_{\mathrm{sp}} L(T_{I\times J}) \max_{b\in P}\|A_b\|_2.$$

The estimate

$$\max_{b\in P}\|A_b\|_2 \le \max_{b\in P}\|B_b\|_2 \le \|B\|_2$$

gives the second part of the assertion. □

The computation of the Frobenius norm of $A \in \mathscr{H}(T_{I\times J},k)$ can be done using (2.7) and (1.4) with at most $c_{\mathrm{sp}} \max\{k^2, n_{\min}\}[|I|\log|I| + |J|\log|J|]$ arithmetic operations. The spectral norm of A can also be computed with logarithmic-linear complexity, for instance, by the power method applied to $A^H A$, which A enters only through the matrix-vector product.

2.5 Adding $\mathscr{H}$-Matrices

The sum of two matrices $A,B \in \mathscr{H}(T_{I\times J},k)$ will usually be in $\mathscr{H}(T_{I\times J},2k)$ but not in $\mathscr{H}(T_{I\times J},k)$. The reason for this is that $\mathbb{C}_k^{m\times n}$ is not a linear space as we have seen in Sect. 1.1.5. Hence, $\mathscr{H}(T_{I\times J},k)$ is not a linear space and we have to approximate the sum $A+B$ by a matrix $S \in \mathscr{H}(T_{I\times J},k)$ if we want to avoid that the rank and hence the complexity grows with each addition. Obviously, this can be done using the rounded addition from Sect. 1.1.5 on each admissible block. On non-admissible block, the usual (entrywise) addition is employed. The addition of $\mathscr{H}$-matrices can therefore easily be parallelized using the scheduling algorithms from Sect. 2.3.2.1.

Since the rounded addition gives a blockwise best approximation (see Theorem 1.7), S is a best approximation in the Frobenius norm

$$\|A+B-S\|_F \leq \|A+B-M\|_F \quad \text{for all } M \in \mathcal{H}(T_{I\times J},k).$$

Using Theorem 2.16, this estimate for the spectral norm reads

$$\|A+B-S\|_2 \leq c_{\mathrm{sp}} L(T_{I\times J})\|A+B-M\|_2 \quad \text{for all } M \in \mathcal{H}(T_{I\times J},k).$$

The following bound on the number of arithmetic operations results from Theorem 1.7, (1.35), and (1.36).

Theorem 2.17. *Let $A,B \in \mathcal{H}(T_{I\times J},k)$. The number of operations required for computing a matrix $S \in \mathcal{H}(T_{I\times J},k)$ satisfying the above error estimates is of the order*

$$c_{\mathrm{sp}} k^2 [L(T_I)|I| + L(T_J)|J|] + c_{\mathrm{sp}} k^3 \min\{|T_I|,|T_J|\}.$$

Alternatively, not k but the blockwise accuracy ε of the approximation can be prescribed; i.e.,

$$\|(A+B)_b - S_b\|_2 \leq \varepsilon \|(A+B)_b\|_2 \quad \text{for all } b \in P.$$

In this case, the required blockwise rank depends on the matrices and cannot be predicted without deeper knowledge of the underlying problem. Using Theorem 2.16, we obtain the relative error estimate

$$\|A+B-S\|_2 \leq c_{\mathrm{sp}} L(T_{I\times J})\varepsilon \|A+B\|_2.$$

2.5.1 Preserving Positivity

In the rest of this chapter we will also define replacements for other common matrix operations. Since these substitutes will all be based on the rounded addition, the introduced error will propagate and will in particular perturb the eigenvalues of the results of these operations. If the smallest eigenvalue is close to the origin compared with the rounding accuracy ε, it may happen that the result of these operations becomes indefinite although it should be positive definite in exact arithmetic. Such a situation can be avoided by the following ideas; cf. [28].

Assume that $\hat{A} \in \mathbb{C}^{I\times I}$ is the Hermitian positive definite result of an exact addition of two matrices from $\mathcal{H}(T_{I\times I},k)$ and let $A \in \mathcal{H}(T_{I\times I},k)$ be its $\mathcal{H}$-matrix approximant. For a moment we assume that $\hat{A}$ and A differ only on a single off-diagonal block $t\times s \in P$. Let EF^H, $E \in \mathbb{C}^{t\times k}$, $F \in \mathbb{C}^{s\times k}$, be the error matrix associated with $t\times s$; i.e.,

$$A_{ts} = \hat{A}_{ts} - EF^H$$

and let

$$\varepsilon := \max\{\|E\|_2^2, \|F\|_2^2\}. \tag{2.10}$$

Due to symmetry, FE^H is the error matrix on block $s \times t$.

We modify the approximant A in such a manner that the new approximant $\tilde{A}$ can be guaranteed to be positive definite. This is done by adding EE^H to A_{tt} and FF^H to A_{ss} such that

$$\begin{bmatrix} \tilde{A}_{tt} & \tilde{A}_{ts} \\ \tilde{A}_{ts}^H & \tilde{A}_{ss} \end{bmatrix} := \begin{bmatrix} A_{tt} & A_{ts} \\ A_{ts}^H & A_{ss} \end{bmatrix} + \begin{bmatrix} EE^H & \\ & FF^H \end{bmatrix} = \begin{bmatrix} \hat{A}_{tt} & \hat{A}_{ts} \\ \hat{A}_{ts}^H & \hat{A}_{ss} \end{bmatrix} + \begin{bmatrix} EE^H & -EF^H \\ -FE^H & FF^H \end{bmatrix}.$$

Since

$$\begin{bmatrix} EE^H & -EF^H \\ -FE^H & FF^H \end{bmatrix} = \begin{bmatrix} -E \\ F \end{bmatrix} \begin{bmatrix} -E \\ F \end{bmatrix}^H$$

is positive semi-definite, the eigenvalues of $\tilde{A}$ are not smaller than those of $\hat{A}$. Therefore, $\tilde{A}$ is Hermitian positive definite and

$$\| \begin{bmatrix} \tilde{A}_{tt} & \tilde{A}_{ts} \\ \tilde{A}_{ts}^H & \tilde{A}_{ss} \end{bmatrix} - \begin{bmatrix} \hat{A}_{tt} & \hat{A}_{ts} \\ \hat{A}_{ts}^H & \hat{A}_{ss} \end{bmatrix} \|_2 = \| \begin{bmatrix} EE^H & -EF^H \\ -FE^H & FF^H \end{bmatrix} \|_2 \leq \|E\|_2^2 + \|F\|_2^2 \leq 2\varepsilon.$$

If a relative error is preferred, we have to guarantee that

$$\|E\|_2^2 \leq \varepsilon \|\hat{A}_{tt}\|_2, \quad \|F\|_2^2 \leq \varepsilon \|\hat{A}_{ss}\|_2 \quad \text{and} \quad \|EF^H\|_2 \leq \varepsilon \|\hat{A}_{ts}\|_2$$

holds instead of (2.10). In this case, we obtain from Theorem 2.16 that

$$\| \begin{bmatrix} \tilde{A}_{tt} & \tilde{A}_{ts} \\ \tilde{A}_{ts}^H & \tilde{A}_{ss} \end{bmatrix} - \begin{bmatrix} \hat{A}_{tt} & \hat{A}_{ts} \\ \hat{A}_{ts}^H & \hat{A}_{ss} \end{bmatrix} \|_2 = \| \begin{bmatrix} EE^H & -EF^H \\ -FE^H & FF^H \end{bmatrix} \|_2 \leq 2\varepsilon \| \begin{bmatrix} \hat{A}_{tt} & \hat{A}_{ts} \\ \hat{A}_{ts}^H & \hat{A}_{ss} \end{bmatrix} \|_2.$$

Since $t \times t$ and $s \times s$ will usually not be leaves in $T_{I \times I}$, it is necessary that EE^H and FF^H are restricted to the leaves of $t \times t$ and $s \times s$ when adding them to A_{tt} and A_{ss}, respectively. Note that this leads to a rounding error which in turn has to be added to the diagonal sub-blocks of $t \times t$ and $s \times s$ in order to preserve positivity. The computational complexity which is connected with the rounded addition makes it necessary to improve the above idea. Once again, we replace an approximant with another approximant by adding a positive semi-definite matrix. Let t_1 and t_2 be the sons of t and let s_1 and s_2 be the sons of s. If we define

$$\tilde{\tilde{A}}_{tt} := \tilde{A}_{tt} + \begin{bmatrix} -E_{t_1} \\ E_{t_2} \end{bmatrix} \begin{bmatrix} -E_{t_1} \\ E_{t_2} \end{bmatrix}^H = A_{tt} + 2 \begin{bmatrix} E_{t_1} E_{t_1}^H & 0 \\ 0 & E_{t_2} E_{t_2}^H \end{bmatrix},$$

the problem of adding EE^H to A_{tt} is reduced to adding $2E_{t_1}E_{t_1}^H$ to $A_{t_1 t_1}$ and $2E_{t_2}E_{t_2}^H$ to $A_{t_2 t_2}$. Applying this idea recursively, adding EE^H to A_{tt} can finally be done by adding a multiple of $E_{t'}E_{t'}^H$ to the dense matrix block $A_{t't'}$ for each leaf t' in T_I from the set of descendants of t. We obtain the following two algorithms addsym_stab and addsym_diag.

procedure addsym_stab$(t,s,U,V,\mathrm{var}A)$
if $t\times s$ is non-admissible **then**
    add UV^H to A_{ts} without approximation;
else
    add UV^H to A_{ts} using the rounded addition;
    denote by EF^H the rounding error;
    addsym_diag(t,E,A);
    addsym_diag(s,F,A);
endif

Algorithm 2.3: Stabilized Hermitian rounded addition.

The first adds a matrix of low rank UV^H to an off-diagonal block $t\times s$ while the latter adds EE^H to the diagonal block $t\times t$. Note that we assume that an Hermitian matrix is represented by its upper triangular part only.

procedure addsym_diag$(t,E,\mathrm{var}A)$
if $t\times t$ is a leaf **then**
    add EE^H to A_{tt} without approximation;
else
    addsym_diag$(t_1,\sqrt{2}E_{t_1},A)$;
    addsym_diag$(t_2,\sqrt{2}E_{t_2},A)$;
endif

Algorithm 2.4: Stabilized diagonal addition.

We will now estimate the costs if the above algorithms are applied to $t\times s\in T_{I\times I}$. Denote by $N^{\mathrm{stab}}_{\mathrm{diag}}(t)$ the number of operations needed if Algorithm 2.4 is applied to $t\in T_I\setminus\mathscr{L}(T_I)$ with $E\in\mathbb{C}^{t\times k}$. Since

$$N^{\mathrm{stab}}_{\mathrm{diag}}(t)=N^{\mathrm{stab}}_{\mathrm{diag}}(t_1)+N^{\mathrm{stab}}_{\mathrm{diag}}(t_2)$$

and since at most $k|t'|^2$ operations are required on each leaf t' of T_t, we obtain

$$N^{\mathrm{stab}}_{\mathrm{diag}}(t)=\sum_{t'\in\mathscr{L}(T_t)}N^{\mathrm{stab}}_{\mathrm{diag}}(t')\le\sum_{t'\in\mathscr{L}(T_t)}k|t'|^2\le n_{\min}k\sum_{t'\in\mathscr{L}(T_t)}|t'|=n_{\min}k\,|t|.$$

Additionally, denote by $N^{\mathrm{stab}}_{\mathrm{add}}(t,s)$ the number of operations required to add $UV^H\in\mathbb{C}^{t\times s}_k$ to A_{ts} using Algorithm 2.3. If $t\times s$ is non-admissible, then $\min\{|t|,|s|\}\le n_{\min}$, which leads to

$$N^{\mathrm{stab}}_{\mathrm{add}}(t,s)\le|t||s|\le n_{\min}(|t|+|s|).$$

Since for admissible $t\times s\in T_{I\times I}$ a rounded addition and two calls to addsym_diag have to be performed, the costs of Algorithm 2.4 can be estimated by

$$\begin{aligned}N^{\mathrm{stab}}_{\mathrm{add}}(t,s)&=\max\{k^2,n_{\min}\}(|t|+|s|)+N^{\mathrm{stab}}_{\mathrm{diag}}(t)+N^{\mathrm{stab}}_{\mathrm{diag}}(s)\\&\le[\max\{k^2,n_{\min}\}+n_{\min}k](|t|+|s|).\end{aligned}$$

Hence, the stabilized addition has asymptotically the same computational complexity as the rounded addition on each block.

If two $\mathscr{H}$-matrices are to be added, the stabilized addition has to be applied to each block. The resulting $\mathscr{H}$-matrix will differ from the result S of the approximate addition from Sect. 2.5 only in the diagonal blocks of P. Hence, it requires the same amount of storage. The following theorem gathers the estimates of this section.

Theorem 2.18. *Let A, B be Hermitian and let λ_i, $i \in I$, denote the eigenvalues of $A+B$. Assume that $S \in \mathscr{H}(T_{I\times I}, k)$ has precision ε. Using the stabilized rounded addition on each block leads to a matrix $\tilde{S} \in \mathscr{H}(T_{I\times I}, k)$ with eigenvalues $\tilde{\lambda}_i \geq \lambda_i$, $i \in I$, satisfying*

$$\|A+B-\tilde{S}\|_2 \sim L(T_I)|I|\varepsilon.$$

Hence, if $A+B$ is positive definite, so is $\tilde{S}$. At most $(\max\{k^2, n_{\min}\}+n_{\min}k)L(T_I)|I|$ operations are required for the construction of $\tilde{S}$.

Proof. Let $t \in T_I^{(\ell)}$ be a cluster from the ℓth level of T_I. Since at most c_{sp} blocks $t \times s$, $s \in T_I$, are contained in P, addsym_diag is applied to t only c_{sp} times during the stabilized addition of A and B. This routine adds terms $2^p E_{t'} E_{t'}^H$, $p < L(T_I) - \ell$, to $S_{t't'}$. Hence, the error on $t' \times t'$ is bounded by

$$c_{\mathrm{sp}} \sum_{\ell=0}^{L(T_I)} 2^{L(T_I)-1-\ell}\varepsilon \leq c_{\mathrm{sp}} 2^{L(T_I)}\varepsilon \leq c\, c_{\mathrm{sp}}|I|\varepsilon$$

with some constant $c > 0$. The previous estimate follows from the fact that the depth $L(T_I)$ of T_I scales like $\log_2|I|$. Since all other blocks blocks coincide with the blocks of S, which have accuracy ε, we obtain the estimate

$$\|A+B-\tilde{S}\|_2 \leq c\, c_{\mathrm{sp}}^2 L(T_{I\times I})|I|\varepsilon \leq c\, c_{\mathrm{sp}}^2 L(T_I)|I|\varepsilon$$

due to Theorem 2.16. □

The stabilized addition will be used in Sect. 3.6.3 when computing approximations to Cholesky decompositions of almost singular matrices.

2.6 Coarsening $\mathscr{H}$-Matrices

In this section we describe how a given matrix $A \in \mathscr{H}(T_{I\times J}, k)$ is approximated by a matrix $\tilde{A} \in \mathscr{H}(T'_{I\times J}, k')$, where $T'_{I\times J}$ is a sub-tree of $T_{I\times J}$ with the same root $I \times J$. In the first part of this section, $k' \leq k$ will be a given number such that the accuracy of $\tilde{A}$ can be estimated only relatively to the best approximation. In the second part we prescribe the accuracy and estimate the resulting rank k'.

We have already got to know the following two coarsening techniques.

(a) *Blockwise coarsening*: Approximants of lower accuracy compared with the accuracy of A are, for instance, sufficient when generating preconditioners of A. In order to improve the data-sparsity and thereby the efficiency of the $\mathscr{H}$-matrix

approximant, it is helpful to remove superfluous information from the blocks. In Sect. 1.1.3 it was described how to compute a low-rank approximant of prescribed accuracy and minimal rank to a given low-rank matrix. Especially in the case of non-local operators (see Chap. 3), this recompression technique is likely to improve the storage requirements even if the accuracy is not reduced. The reason for this is that low-rank approximations are usually generated from non-optimal constructions.

(b) *Agglomeration of blocks*: The partition P generated in Sect. 1.3 is admissible and can be computed with logarithmic-linear complexity. We have remarked that P, however, may be non-optimal. Hence, there is a good chance to improve it by agglomerating blocks using the procedure from Sect. 1.1.6; see also [115]. The agglomeration of blocks will be particularly beneficial to the efficiency of arithmetic operations such as multiplication and inversion of $\mathscr{H}$-matrices due to an improved sparsity constant.

Coarsening can be applied to a whole $\mathscr{H}$-matrix but also to a submatrix. Without loss of generality we consider only the case that $T'_{I\times J} = I \times J$; i.e., A is coarsened to a single matrix block $\tilde{A}$ of rank k'.

2.6.0.1 Coarsening with Prescribed Rank

For the first part of this section assume that $k' \le k$ is given. We start from the tree $T_L := T_{I\times J}$, $L := L(T_{I\times J}) - 1$, and a matrix $A_L \in \mathscr{H}(T_L, k')$ which is generated from A by approximating each block A_b, $b \in \mathscr{L}(T_{I\times J})$, by a matrix of rank k' using the technique from Sect. 1.1.3. This first coarsening step requires

$$\sum_{t\times s\in\mathscr{L}(T_{I\times J})} 6k^2(|t|+|s|)+20k^3 \le 6c_{\mathrm{sp}}k^2L(T_{I\times J})[|I|+|J|]+40c_{\mathrm{sp}}k^3\min\{|I|,|J|\}/n_{\min}$$

arithmetic operations due to (1.36) and Lemma 1.39. Since A is approximated on each block by a best approximation, for the Frobenius norm it holds that

$$\|A-A_L\|_F \le \|A-M\|_F \quad \text{for all } M \in \mathscr{H}(T_L,k'). \tag{2.11}$$

The following rule defines a sequence of block cluster trees T_ℓ and an associated sequence of approximants $A_\ell \in \mathscr{H}(T_\ell,k')$, $\ell = L-1,\dots,0$. Let T_ℓ result from $T_{\ell+1}$ by removing the sons of each block $b \in T_{\ell+1}^{(\ell)} \setminus \mathscr{L}(T_{\ell+1})$ in the ℓth level of $T_{\ell+1}$. A_ℓ results from $A_{\ell+1}$ by the agglomeration procedure from Sect. 1.1.6 applied to such blocks b. The property that a best approximation is attained on each block is inherited by the whole matrix with respect to the Frobenius norm; i.e,

$$\|A_{\ell+1}-A_\ell\|_F \le \|A_{\ell+1}-M\|_F \quad \text{for all } M \in \mathscr{H}(T_\ell,k'). \tag{2.12}$$

In order to agglomerate a non-admissible block, it is first converted to the outer-product representation using the SVD.

The number of arithmetic operations of the above construction is determined by the number of operations required for the SVD of each non-admissible block and the numerical effort of the agglomeration of each block $b \in T_{I\times J} \setminus \mathscr{L}(T_{I\times J})$. According to Sect. 1.1.6, at most

$$\sum_{t\times s\in T_{I\times J}} 24k^2(|t|+|s|) + \max\{1408\frac{2}{3}k^3, 22n_{\min}^3\}$$

operations are required, where we have used that $\max\{|t|,|s|\} \le n_{\min}$, which may be assumed due to the same level of a block's row and column cluster. Using (1.35) we obtain

Lemma 2.19. *The number of arithmetic operations required for the above construction of A_0 is of the order*

$$c_{\mathrm{sp}}k^2L(T_{I\times J})(|I|+|J|) + \max\{k^3, n_{\min}^3\}|T_{I\times J}|.$$

Using the above procedure, $A \in \mathscr{H}(T_{I\times J},k)$ is rounded to a matrix of rank k'. The resulting approximation error can be arbitrarily bad. Assume we know a best approximant $A_{\mathrm{best}} \in \mathbb{C}_{k'}^{I\times J}$ for A. The following lemma (cf. [116]) relates the approximation error $\|A-A_0\|_F$ to the best possible error $\|A-A_{\mathrm{best}}\|_F$.

Lemma 2.20. *Let $A_0 \in \mathbb{C}_{k'}^{I\times J}$ be constructed as above. Then it holds that*

$$\|A-A_0\|_F \le 2^{L(T_{I\times J})}\|A-A_{\mathrm{best}}\|_F.$$

Proof. Let $L = L(T_{I\times J}) - 1$. From (2.12) it follows that

$$\|A_\ell - A_{\mathrm{best}}\|_F \le \|A_\ell - A_{\ell+1}\|_F + \|A_{\ell+1} - A_{\mathrm{best}}\|_F \le 2\|A_{\ell+1} - A_{\mathrm{best}}\|_F$$

for $\ell \in \{0,\dots,L-1\}$, because $A_{\mathrm{best}} \in \mathscr{H}(T_\ell,k')$. Hence, $\|A_\ell - A_{\mathrm{best}}\|_F \le 2^{L-\ell}\|A_L - A_{\mathrm{best}}\|_F$ and we obtain from (2.12)

$$\begin{aligned}\|A_L - A_0\|_F = \|\sum_{\ell=0}^{L-1}(A_\ell - A_{\ell+1})\|_F &\le \sum_{\ell=0}^{L-1}\|A_\ell - A_{\ell+1}\|_F \le \sum_{\ell=0}^{L-1}\|A_{\ell+1} - A_{\mathrm{best}}\|_F \\ &\le \sum_{\ell=0}^{L-1} 2^{L-\ell-1}\|A_L - A_{\mathrm{best}}\|_F = (2^L-1)\|A_L - A_{\mathrm{best}}\|_F.\end{aligned}$$

The assertion follows from

$$\begin{aligned}\|A-A_0\|_F &\le \|A-A_L\|_F + \|A_L - A_0\|_F \le \|A-A_L\|_F + (2^L-1)\|A_L - A_{\mathrm{best}}\|_F \\ &\le 2^L\|A-A_L\|_F + (2^L-1)\|A-A_{\mathrm{best}}\|_F \le (2^{L+1}-1)\|A-A_{\mathrm{best}}\|_F\end{aligned}$$

due to (2.11). □

2.6.0.2 Coarsening with Prescribed Accuracy

If on the other hand a given accuracy $\varepsilon > 0$ is to be satisfied in each agglomeration step, i.e., we have

$$\|A_{\ell+1} - A_\ell\|_F \leq \varepsilon \|A_{\ell+1}\|_F, \quad 0 \leq \ell < L, \tag{2.13}$$

then the required rank k' may increase. In order to avoid that the complexity is deteriorated, we stop the coarsening process in blocks for which the required rank k' is such that agglomeration is not worthwhile.

Assume that the sub-blocks $S(b)$ of a block $b = t \times s \in T_{\ell+1}^{(\ell)} \setminus \mathscr{L}(T_{\ell+1})$ in $A_{\ell+1}$ are low-rank matrices with ranks $k_{t'\times s'}$, $t' \times s' \in S(b)$. By comparing the original storage costs of $(A_{\ell+1})_b$ with those of $(A_\ell)_b$, it is easy to check whether the coarsening leads to reduced costs of the approximant. If

$$k_{t\times s}(|t| + |s|) \leq \sum_{t'\times s' \in S(b)} k_{t'\times s'}(|t'| + |s'|), \tag{2.14}$$

then the block cluster tree $T_{\ell+1}$ is modified by replacing the sons $S(b)$ of b by the new leaf b. If this condition is not satisfied, then the sons of b will be kept in the block cluster tree. This procedure can then be applied to the leaves of the new block cluster tree until (2.14) is not satisfied.

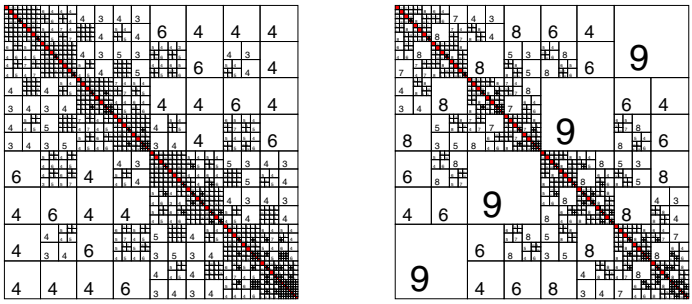


Fig. 2.9 $\mathscr{H}$-matrix before and after coarsening.

It is obvious that this procedure does not increase the amount of storage. In contrast, depending on A it will usually improve the storage requirements. In the following two lemmas (see [28]) we analyze the accuracy and the complexity of this adaptive agglomeration process. For this purpose we consider a single block $b \in T_{I\times J}$ in which the described agglomeration stops due to the violation of (2.14) by the father block of b. Without loss of generality we may identify b with $I \times J$ and assume that (2.14) holds for all $t \times s \in T_{I\times J} \setminus \mathscr{L}(T_{I\times J})$. The following lemma describes the accuracy of A_0 compared with the accuracy of A_L.

Lemma 2.21. *Let $A \in \mathbb{C}^{I\times J}$. If $\|A-A_L\|_F \le \varepsilon\|A\|_F$, then $\|A-A_0\|_F \le c(\varepsilon)\|A\|_F$, where $c(\varepsilon) = \varepsilon + (1+\varepsilon)[(1+\varepsilon)^L - 1] \sim L(T_{I\times J})\varepsilon$ for $\varepsilon \to 0$.*

Proof. Due to $\|A_\ell\|_F \le \|A_{\ell+1}\|_F + \|A_{\ell+1}-A_\ell\|_F \le (1+\varepsilon)\|A_{\ell+1}\|_F$, from (2.13) we have that

$$\begin{aligned}\|A_L - A_0\|_F &= \|\sum_{\ell=0}^{L-1}(A_{\ell+1}-A_\ell)\|_F \le \sum_{\ell=0}^{L-1}\|A_{\ell+1}-A_\ell\|_F \le \varepsilon\sum_{\ell=0}^{L-1}\|A_{\ell+1}\|_F \\ &\le \varepsilon\sum_{\ell=0}^{L-1}(1+\varepsilon)^{L-\ell-1}\|A_L\|_F = [(1+\varepsilon)^L - 1]\|A_L\|_F.\end{aligned}$$

Observing

$$\begin{aligned}\|A-A_0\|_F &\le \|A-A_L\|_F + \|A_L-A_0\|_F \le \varepsilon\|A\|_F + [(1+\varepsilon)^L-1]\|A_L\|_F \\ &\le \varepsilon\|A\|_F + [(1+\varepsilon)^L-1](\|A-A_L\|_F + \|A\|_F) \\ &\le \left\{\varepsilon + (1+\varepsilon)[(1+\varepsilon)^L-1]\right\}\|A\|_F,\end{aligned}$$

we obtain the assertion. □

The computational cost of the coarsening procedure is estimated in the following lemma.

Lemma 2.22. *The resulting rank of A_0 is bounded by $c_{\mathrm{sp}}k_{\max}L(T_{I\times J})$. Hence, the required costs are of the order*

$$c_{\mathrm{sp}}^3 k_{\max}^2 L^3(T_{I\times J})[|I|+|J|],$$

where $k_{\max} := \max_{t\times s\in\mathscr{L}(T_{I\times J})} k_{t\times s}$.

Proof. Similarly to the case of a blockwise constant rank, the costs of coarsening $T_{I\times J}$ can be estimated to be bounded by

$$\sum_{t\times s\in T_{I\times J}} k_{t\times s}^2(|t|+|s|),$$

where we have omitted terms which do depend neither on $|t|$ nor on $|s|$. Using (2.14), the cost of each block $t\times s \in T_{I\times J}\setminus\mathscr{L}(T_{I\times J})$ can be estimated by a sum over its leaves

$$k_{t\times s}(|t|+|s|) \le \sum_{t'\times s'\in\mathscr{L}(T_{t\times s})} k_{t'\times s'}(|t'|+|s'|).$$

From (1.36) it follows that $k_{t\times s} \le c_{\mathrm{sp}}L(T_{t\times s})k_{\max}$. With the previous estimate we obtain

$$\begin{aligned}\sum_{t\times s\in T_{I\times J}} k_{t\times s}^2(|t|+|s|) &\le c_{\mathrm{sp}}^2 L^2(T_{I\times J})k_{\max}^2 \sum_{t\times s\in T_{I\times J}}(|t|+|s|) \\ &\le c_{\mathrm{sp}}^3 L^3(T_{I\times J})k_{\max}^2[|I|+|J|]\end{aligned}$$

due to (1.36). □

We have seen that each block $t \times s \in \mathscr{L}(T'_{I\times J})$ of the final partition $\mathscr{L}(T'_{I\times J})$ requires $\mathscr{O}(k_{\max}^2(|t|+|s|))$ operations for its computation and that its accuracy is of the order $L(T_{I\times J})\varepsilon$. Returning to the whole matrix, the coarsening process therefore gives back a matrix $\tilde{A}$ which has accuracy $L(T_{I\times J})\varepsilon$ and can be computed with $\mathscr{O}(k_{\max}^2(|I|+|J|))$ arithmetic operations; cf. (1.36).

2.7 Multiplying $\mathscr{H}$-Matrices

Let $A \in \mathscr{H}(T_{I\times J}, k_A)$ and $B \in \mathscr{H}(T_{J\times K}, k_B)$ be two hierarchical matrices. The aim of this section is to investigate the product $AB \in \mathbb{C}^{I\times K}$ of A and B. In contrast to the hierarchical addition, which preserves the block structure, the exact product AB cannot be represented using the block cluster tree $T_{I\times K}$, i.e., it cannot be guaranteed that the blockwise rank is bounded in general. Therefore, in the first part of this section we will define the product tree T_{IJK}, which is suitable to hold the exact product AB. The second part of this section is devoted to the rounded multiplication to a given partition and given rank. Our presentation mainly relies on [116].

2.7.1 Product Block Cluster Tree

In this section we assume that $T_{I\times J}$ has been generated using the cluster trees T_I and T_J and that for the construction of $T_{J\times K}$ the cluster trees T_J and T_K have been used.

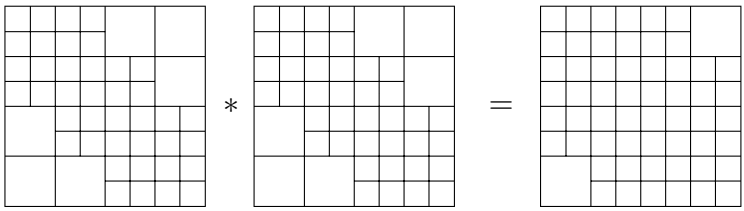


Fig. 2.10 The product of two partitions.

Figure 2.10 shows that the block structure is usually not preserved. The block in the upper right corner of the product is a sum of products in which at least one factor is a low-rank matrix. Hence, its rank is bounded by the number of products times the maximum rank of blocks in A and B. The block on the left of the latter needs to be refined since for its computation a product of two factors which are not in the set of leaves is involved. The impression that the product partition is always finer than the partition of the factors is wrong as can be seen from Fig. 2.11.

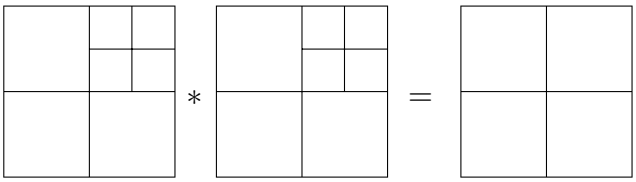


Fig. 2.11 Factors leading to a coarser product partition.

Definition 2.23. The **product tree** T_{IJK} of $T_{I\times J}$ and $T_{J\times K}$ is inductively defined by

(i) $I\times K$ is the root of T_{IJK}
(ii) The set of sons of blocks $t\times s\in T_{IJK}$ from the ℓth level of T_{IJK} is

$$S_{IJK}(t\times s) := \Big\{ t'\times s' \,|\, \exists r\in T_J^{(\ell)}, r'\in T_J^{(\ell+1)} : t'\times r'\in S_{I\times J}(t\times r) \\ \text{and } r'\times s'\in S_{J\times K}(r\times s)\Big\}.$$

Lemma 2.24. *The product tree T_{IJK} is a block cluster tree based on T_I and T_K. For its depth it holds that*

$$L(T_{IJK}) \le \min\{L(T_{I\times J}), L(T_{J\times K})\}.$$

The sparsity constant of T_{IJK} can be estimated as

$$c_{\mathrm{sp}}(T_{IJK}) \le c_{\mathrm{sp}}(T_{I\times J})\cdot c_{\mathrm{sp}}(T_{J\times K}).$$

Proof. Due to Definition 2.23 it holds that for a given $t\in T_I$

$$\{s\in T_K : t\times s\in T_{IJK}\} \subset \{s\in T_K \,|\, \exists r\in T_J : t\times r\in T_{I\times J} \text{ and } r\times s\in T_{J\times K}\}.$$

Therefore, recalling Definition 1.35 we have that

$$\begin{aligned} |\{s\in T_K : t\times s\in T_{IJK}\}| &\le \sum_{r\in T_J : t\times r\in T_{I\times J}} |\{s\in T_K : r\times s\in T_{J\times K}\}| \\ &\le c_{\mathrm{sp}}(T_{I\times J})\cdot c_{\mathrm{sp}}(T_{J\times K}). \end{aligned}$$

The rest of the assertion is an easy consequence of Definition 2.23. □

Let $t\times s\in T_{IJK}$ be a leaf from the ℓth level of T_{IJK}. Then

$$(AB)_{ts} = \sum_{j\in J} A_{tj}B_{js}.$$

We will rearrange the previous summation to a sum of products in which one factor is a low-rank matrix. To this end, denote by $F_j(t)\in T_I$ and $F_j(s)\in T_K$ the uniquely defined ancestors of t and s from the jth, $0\le j\le \ell$, level of T_I and T_K, respectively. We define the set

$$U_j(t\times s) := \Big\{ r \in T_J^{(j)} : F_j(t)\times r \in T_{I\times J} \text{ and } r\times F_j(s) \in \mathscr{L}(T_{J\times K}) \\ \text{or } F_j(t)\times r \in \mathscr{L}(T_{I\times J}) \text{ and } r\times F_j(s) \in T_{J\times K} \Big\}.$$

The consequence of the following lemma is that

$$(AB)_{ts} = \sum_{j=0}^{\ell} \sum_{r\in U_j(t\times s)} A_{tr}B_{rs}. \tag{2.15}$$

Lemma 2.25. *It holds that*

$$\bigcup_{j=0}^{\ell} \bigcup_{r\in U_j(t\times s)} r = J,$$

where the union is pairwise disjoint. Furthermore, it holds that

$$|U_j(t\times s)| \le \min\{c_{\mathrm{sp}}(T_{I\times J}), c_{\mathrm{sp}}(T_{J\times K})\}, \quad 0\le j\le \ell.$$

Proof. Let $\nu\in J$. It holds that $b_0 := F_0(t)\times J \in T_{I\times J}$ and $b_0' := J\times F_0(s)\in T_{J\times K}$. If neither b_0 nor b_0' is a leaf, then there is $r_1\in S_J(J)$ such that $\nu\in r_1$ and $b_1 := F_1(t)\times r_1 \in T_{I\times J}$, $b_1' := r_1\times F_1(s)\in T_{J\times K}$. If still neither b_1 nor b_1' is a leaf, we descend the trees keeping $\nu\in r_j$ until for some j either $b_j := F_j(t)\times r_j$ or $b_j' := r_j\times F_j(s)$ is a leaf. In this case $\nu\in r_j\in U_j$. Since $t\times s$ is a leaf in T_{IJK}, it follows that $j\le\ell$.

Since U_j is constructed from $T_J^{(j)}$, the elements of each U_j are pairwise disjoint. Let $r\in U_j$ and $r'\in U_{j'}$, $j\le j'$, and $r\cap r'\neq\varnothing$. Since $r,r'\subset T_J$, we obtain $r'\subset r$. It follows that

$$F_{j'}(t)\times r' \subset F_j(t)\times r \quad \text{and} \quad r'\times F_{j'}(s)\subset r\times F_j(s). \tag{2.16}$$

The definition of U_j implies that either $F_j(t)\times r$ or $r\times F_j(s)$ is a leaf. Hence, one of the inclusions in (2.16) is an equality, which implies that $j=j'$ and hence that $r=r'$.

From

$$|U_j| \le |\{r\in T_J^{(j)} : F_j(t)\times r\in T_{I\times J}\}| \le c_{\mathrm{sp}}(T_{I\times J})$$

and

$$|U_j| \le |\{r\in T_J^{(j)} : r\times F_j(s)\in T_{J\times K}\}| \le c_{\mathrm{sp}}(T_{J\times K})$$

we obtain the estimate on the cardinality of U_j. □

Theorem 2.26. *Let $L=L(T_{IJK})$. For the product AB of two matrices $A\in\mathscr{H}(T_{I\times J},k_A)$ and $B\in\mathscr{H}(T_{J\times K},k_B)$ it holds that $AB\in\mathscr{H}(T_{IJK},k)$, where*

$$k \le L\min\{c_{\mathrm{sp}}(T_{I\times J}), c_{\mathrm{sp}}(T_{J\times K})\}\max\{k_A,k_B,n_{\min}\}.$$

The matrix AB can be computed with at most

$$c_{\mathrm{sp}}(T_{IJK})L\max\{k_B'N_{\mathrm{MV}}(A), k_A'N_{\mathrm{MV}}(B)\}$$

arithmetic operations. Here, $N_{\mathrm{MV}}(A)$ denotes the number of arithmetic operations required for the matrix-vector multiplication (see Sect. 2.2), $k'_A := \max\{k_A, n_{\min}\}$, and $k'_B := \max\{k_B, n_{\min}\}$.

Proof. Let $t \times s \in \mathscr{L}(T_{IJK})$ be from the ℓth level of T_{IJK}. Due to (2.15) we can express $(AB)_{ts}$ by the sum over $L \max_{j=0,\dots,\ell} |U_j(t \times s)|$ matrix products. From the definition of $U_j(t \times s)$ and from $t \times r \subset F_j(t) \times r$ and $r \times s \subset r \times F_j(s)$ we see that one of the factors of each product corresponds to a leaf and so its rank is bounded by $\max\{k_A, k_B, n_{\min}\}$. As a consequence,

$$k \le L \max_{j=0,\dots,\ell} |U_j(t \times s)| \max\{k_A, k_B, n_{\min}\}.$$

The first part of the assertion follows from Lemma 2.25.

Using the representation (2.15), we have to compute the products $A_{tr}B_{rs}$, each of which consists of $\max\{k_A, k_B, n_{\min}\}$ matrix-vector products. Hence, with $P := \mathscr{L}(T_{IJK})$ and $P_i := P \cap T_{IJK}^{(i)}$, $0 \le i < L$, we obtain for the number of arithmetic operations for the matrix product

$$\begin{aligned}
N_{\mathrm{MM}}(A,B) &\le \sum_{t\times s\in P} \sum_{j=0}^{L-1} \sum_{r\in U_j(t\times s)} \max\{k'_B N_{\mathrm{MV}}(A_{tr}), k'_A N_{\mathrm{MV}}(B_{rs})\} \\
&\le \sum_{t\times s\in P} \max\{k'_B N_{\mathrm{MV}}(A_{tJ}), k'_A N_{\mathrm{MV}}(B_{Js})\} \\
&\le \sum_{i=0}^{L-1} \sum_{t\times s\in P_i} \max\{k'_B N_{\mathrm{MV}}(A_{tJ}), k'_A N_{\mathrm{MV}}(B_{Js})\} \\
&\le c_{\mathrm{sp}}(T_{IJK}) L \max\{k'_B N_{\mathrm{MV}}(A), k'_A N_{\mathrm{MV}}(B)\},
\end{aligned}$$

which proves the assertion. □

2.7.2 Preserving the Original Block Structure

The exact product AB of two $\mathscr{H}$-matrices $A \in \mathscr{H}(T_{I\times J}, k_A)$ and $B \in \mathscr{H}(T_{J\times K}, k_B)$ can be found in the set $\mathscr{H}(T_{IJK}, k')$ with a slightly increased blockwise rank k' as we have just seen in Theorem 2.26. Since the product tree T_{IJK} will usually lead to a finer partition, which in turn leads to an increased numerical effort, the product AB is preferably represented on the usual partition of the cluster tree $T_{I\times K}$ with possibly further increased rank. By the following *idempotency constant* it is possible to estimate this increment. For simplicity we restrict ourselves to the case $I = J = K$.

Definition 2.27. Let $T_{I\times I}$ be a block cluster tree generated from the cluster tree T_I. The **idempotency constant** c_{id} is defined as

$$c_{\mathrm{id}}(b) := \left|\left\{t' \times s' \in T_{I\times I} : t' \times s' \subset b \text{ and } \exists r' \in T_I \text{ with } t' \times r', r' \times s' \in T_{I\times I}\right\}\right|$$

for $b \in \mathscr{L}(T_{I\times I})$ and

$$c_{\mathrm{id}} := \max_{b\in\mathscr{L}(T_{I\times I})} c_{\mathrm{id}}(b).$$

If T_{III} is not finer than $T_{I\times I}$, then $c_{\mathrm{id}} = 1$. In Fig. 2.10 four blocks are refined into its four sons, respectively. In this case it holds that $c_{\mathrm{id}} = 5$.

Example 2.28. In Example 1.36 we have estimated the sparsity constant c_{sp} under the assumption (see (1.22)) that

$$(\operatorname{diam} X_t)^m \le c_g 2^{-\ell} \quad \text{and} \quad \mu(X_t) \ge 2^{-\ell}/c_G$$

for all $t \in T_I^{(\ell)}$. The same assumption will now be used to estimate the idempotency constant. Let $b \in \mathscr{L}(T_{I\times I})$ be from the ℓth level of $T_{I\times I}$. If b is a non-admissible leaf, then $c_{\mathrm{id}}(b) = 1$. For admissible $b = t\times s$ we define

$$q := \log_2(c_g c_G c_\Omega) + m\log_2(2+\eta),$$

where c_Ω is defined in (1.19). We will show that for clusters $t', r', s' \in T_I$, $t'\times s' \subset b = t\times s$, satisfying $t'\times r', r'\times s' \in T_{I\times I}^{(\ell+q)}$ it follows that either $t'\times r'$ or $r'\times s'$ is a leaf in $T_{I\times I}$. This can be seen from

$$2^{-q/m} = c_g^{-1/m} c_G^{-1/m} c_\Omega^{-1/m} (2+\eta)^{-1}$$

and

$$\begin{aligned}
\operatorname{diam} X_{r'} &= (1+\eta/2)\operatorname{diam} X_{r'} - \eta/2 \operatorname{diam} X_{r'} \\
&\le (1+\eta/2) c_g^{1/m} 2^{-(\ell+q)/m} - \eta/2 \operatorname{diam} X_{r'} \\
&\le c_\Omega^{-1/m}/2 \min\{\mu^{1/m}(X_t), \mu^{1/m}(X_s)\} - \eta/2 \operatorname{diam} X_{r'} \\
&\le \frac{1}{2}\min\{\operatorname{diam} X_t, \operatorname{diam} X_s\} - \eta/2 \operatorname{diam} X_{r'} \\
&\le \frac{\eta}{2}(\operatorname{dist}(X_t, X_s) - \operatorname{diam} X_{r'}) \\
&\le \eta \max\{\operatorname{dist}(X_{t'}, X_{r'}), \operatorname{dist}(X_{r'}, X_{s'})\}.
\end{aligned}$$

Since hence either $t'\times r'$ or $r'\times s'$ is admissible, one of these blocks does not have descendants in $T_{I\times I}$. Hence, the number of vertices in T_{III} which are contained in b is bounded by $c_{\mathrm{id}} \le 4^q = (c_g c_G c_\Omega)^2 (2+\eta)^{2m}$.

The same kind of proof can be adapted to partitions generated from algebraic clustering.

Lemma 2.29. *Under the assumption* (1.28) *it holds that* $c_{\mathrm{id}} \le [c_u(2+\eta)]^{2m}$.

Proof. Let $b \in \mathscr{L}(T_{I\times I})$ be from the ℓth level of $T_{I\times I}$. If b is a non-admissible leaf, then $c_{\mathrm{id}}(b) = 1$. For admissible $b = t\times s$ we define

$$q := m \log_2 c_u (2+\eta).$$

We will show that for clusters $t', r', s' \in T_I$, $t' \times s' \subset b = t \times s$, satisfying $t' \times r', r' \times s' \in T_{I\times I}^{(\ell+q)}$ it follows that either $t' \times r'$ or $r' \times s'$ is a leaf in $T_{I\times I}$. This can be seen from

$$\begin{aligned}
\operatorname{diam} r' &= (1+\eta/2)\operatorname{diam} r' - \eta/2 \operatorname{diam} r' \\
&\le (1+\eta/2) c_u 2^{-q/m} \min\{\operatorname{diam} t, \operatorname{diam} s\} - \eta/2 \operatorname{diam} r' \\
&\le \frac{1}{2} \min\{\operatorname{diam} t, \operatorname{diam} s\} - \eta/2 \operatorname{diam} r' \\
&\le \frac{\eta}{2} (\operatorname{dist}(t,s) - \operatorname{diam} r') \\
&\le \eta \max\{\operatorname{dist}(t', r'), \operatorname{dist}(r', s')\}.
\end{aligned}$$

Since hence either $t' \times r'$ or $r' \times s'$ is admissible, one of these blocks does not have descendants in $T_{I\times I}$. Therefore, the number of vertices which are contained in b is bounded by $c_{\mathrm{id}} \le 4^q = [c_u(2+\eta)]^{2m}$. □

Theorem 2.30. *Let $A, B \in \mathscr{H}(T_{I\times I}, k)$. Then $AB \in \mathscr{H}(T_{I\times I}, \hat{k})$, where*

$$\hat{k} \le c_{\mathrm{id}} c_{\mathrm{sp}} L(T_{I\times I}) \max\{k, n_{\min}\}.$$

Proof. Due to Theorem 2.26, we have $AB \in \mathscr{H}(T_{III}, k')$, where the blockwise rank k' is bounded by $c_{\mathrm{sp}} L(T_{I\times I}) \max\{k, n_{\min}\}$. If a leaf from $T_{I\times I}$ is contained in a leaf from T_{III}, then the restriction does not increase the rank. If a leaf from $T_{I\times I}$ contains leaves from T_{III}, then their number is bounded by c_{id}. Therefore, the rank is bounded by $c_{\mathrm{id}} k'$. □

2.7.3 Rounded Multiplication

If the product $AB \in \mathscr{H}(T_{I\times I}, k)$ is to be approximated by a matrix from $\mathscr{H}(T_{I\times I}, \tilde{k})$, $\tilde{k} < k$, then one of the rounding algorithms from Sect. 1.1.5 can be used to reduce the blockwise rank to $\tilde{k}$. The first sums up all arising products in (2.15) and rounds the result to rank $\tilde{k}$. As shown in Sect. 1.1.5, the result of the rounding can be controlled relatively to the best approximation. The second algorithm gradually adds the products to a rounded sum and is significantly faster but can result in arbitrarily large errors for general matrices. The following divide-and-conquer algorithm for the computation of the approximate update $C := C + AB$ of $C \in \mathscr{H}(T_{I\times I}, \tilde{k})$ stems from the blockwise matrix multiplication and is even faster.

Assume that $A \in \mathscr{H}(T_{I\times J}, k_A)$ and $B \in \mathscr{H}(T_{J\times K}, k_B)$ are subdivided according to their block cluster trees $T_{I\times J}$ and $T_{J\times K}$:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}.$$

Then AB has the following block structure

$$AB = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}.$$

If the target matrix C has sons in $T_{I\times K}$, then compute

$$C_{ij} := C_{ij} + A_{i1}B_{1j} + A_{i2}B_{2j}, \quad i,j = 1,2,$$

each of which has approximately half the size of $C := C + AB$. In the case that C is a leaf in $T_{I\times K}$, the sums $A_{i1}B_{1j} + A_{i2}B_{2j}$ are rounded to rank-$\tilde{k}$ matrices R_{ij}, $i,j = 1,2$, and

$$\begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}$$

is agglomerated to a single rank-$\tilde{k}$ matrix (see Sect. 1.1.6) before adding it to C using one of the rounded additions. The complexity of the rounded multiplication was shown to be of the order $k^2L^2(T_I)|I| + k^3|I|$ for $I = J$; see [132]. A parallel version of the previous algorithm was presented in [166].

2.7.4 Multiplication of Hierarchical and Semi-Separable Matrices

As we have seen in Sect. 1.2, multiplying a (p,q)-semi-separable matrix S and a general matrix $A \in \mathbb{C}^{m\times n}$ can be done with $\mathcal{O}((p+q)mn)$ arithmetic operations. If A is an $\mathcal{H}$-matrix, then AS and SA can be computed with significantly less effort. Since diagonal-plus-semi-separable matrices are hierarchical matrices, we could use the hierarchical matrix multiplication algorithm from the previous section. We will however present an algorithm which is significantly more efficient if one of the matrices is semi-separable. Its complexity will actually be of the order of the hierarchical matrix-vector multiplication. The resulting algorithms will be used in Sect. 4.6 when updating the factors of the LU decomposition in Broyden's method.

In the following lemma (see [24]), which is the basis for the efficient multiplication, we make use of the notation

$$s' := \{i \in I : i < \min s\} \quad \text{and} \quad s'' := \{i \in I : i > \max s\}$$

for $s \subset I$. Note that this lemma holds for arbitrary partitions P of the matrix indices $\{1,\dots,m\} \times \{1,\dots,n\}$.

Lemma 2.31. *Let $A \in \mathbb{C}^{m\times n}$ and let $S \in \mathbb{C}^{n\times n}$ be a diagonal-plus-semi-separable matrix with the notation from Definition 1.10. For $t \times s \in P$ it holds that*

$$(AS)_{ts} = A_{ts}S_{ss} + \Xi V_s^H + \Upsilon Z_s^H,$$

where $\Xi := A_{ts'}U_{s'} \in \mathbb{C}^{t\times p}$ and $\Upsilon := A_{ts''}W_{s''} \in \mathbb{C}^{t\times q}$. Furthermore, if $A \in \mathbb{C}^{n\times m}$, then

$$(SA)_{ts} = S_{tt}A_{ts} + U_t\hat{\Xi}^H + W_t\hat{\Upsilon}^H,$$

where $\hat{\Xi} := A^H_{t''s}V_{t''} \in \mathbb{C}^{s\times p}$ *and* $\hat{\Upsilon} := A^H_{t's}Z_{t'} \in \mathbb{C}^{s\times q}$. *Here,* U_t *denotes the restriction of* U *to the rows* t.

Proof. By ℓ we denote the level of $t\times s$ in the block cluster tree $T_{I\times I}$. Let $\tau\times\sigma \supset t\times s$ be in the kth level of $T_{I\times I}$. By induction over $k \le \ell$ we will prove that

$$A_{t\sigma}S_{\sigma s} = A_{ts}S_{ss} + \Xi(\sigma)V_s^H + \Upsilon(\sigma)Z_s^H \tag{2.17}$$

with vectors $\Xi(\sigma) := A_{t\sigma^*}U_{\sigma^*}$ and $\Upsilon(\sigma) = A_{t\sigma^{**}}W_{\sigma^{**}}$, where $\sigma^* := \{i\in\sigma : i < \min s\}$, $\sigma^{**} := \{i\in\sigma : i > \max s\}$. The choice $\sigma = I$ will then lead to the assertion.

For $k = \ell$ we have that $\tau\times\sigma = t\times s$. Therefore, we obtain

$$A_{t\sigma}S_{\sigma s} = A_{ts}S_{ss}.$$

Assume that (2.17) is true for $k+1 \le \ell$. Let $A_{\tau\sigma}$ and $S_{\sigma\sigma}$ be partitioned corresponding to the tree $T_{I\times I}$

$$A_{\tau\sigma} = \begin{bmatrix} A_{\tau_1\sigma_1} & A_{\tau_1\sigma_2} \\ A_{\tau_2\sigma_1} & A_{\tau_2\sigma_2} \end{bmatrix} \quad\text{and}\quad S_{\sigma\sigma} = \begin{bmatrix} S_{\sigma_1\sigma_1} & U_{\sigma_1}V_{\sigma_2}^H \\ W_{\sigma_2}Z_{\sigma_1}^H & S_{\sigma_2\sigma_2} \end{bmatrix}.$$

Then we have that

$$A_{t\sigma}S_{\sigma s} = \begin{bmatrix} A_{\tau_1\sigma_1}S_{\sigma_1\sigma_1} + A_{\tau_1\sigma_2}W_{\sigma_2}Z_{\sigma_1}^H & A_{\tau_1\sigma_2}S_{\sigma_2\sigma_2} + A_{\tau_1\sigma_1}U_{\sigma_1}V_{\sigma_2}^H \\ A_{\tau_2\sigma_1}S_{\sigma_1\sigma_1} + A_{\tau_2\sigma_2}W_{\sigma_2}Z_{\sigma_1}^H & A_{\tau_2\sigma_2}S_{\sigma_2\sigma_2} + A_{\tau_2\sigma_1}U_{\sigma_1}V_{\sigma_2}^H \end{bmatrix}_{ts}.$$

If $s\subset\sigma_1$, then

$$A_{t\sigma}S_{\sigma s} = A_{t\sigma_1}S_{\sigma_1 s} + A_{t\sigma_2}W_{\sigma_2}Z_s^H = A_{ts}S_{ss} + \Xi(\sigma_1)V_s^H + [\Upsilon(\sigma_1) + A_{t\sigma_2}W_{\sigma_2}]Z_s^H.$$

If, on the other hand, $s\subset\sigma_2$, then

$$A_{t\sigma}S_{\sigma s} = A_{t\sigma_2}S_{\sigma_2 s} + A_{t\sigma_1}U_{\sigma_1}V_s^H = A_{ts}S_{ss} + [\Xi(\sigma_2) + A_{t\sigma_1}U_{\sigma_1}]V_s^H + \Upsilon(\sigma_2)Z_s^H$$

due to the induction assumption. The second part of the assertion is obtained by similar arguments. □

According to the previous lemma, each sub-block $(AS)_{ts}$ of AS is a rank-$(p+q)$ update of $A_{ts}S_{ss}$. Let $t_1\times s_1,\dots,t_\mu\times s_\mu$ be the blocks in P such that $\min s_i \le \min s_j$ for $i\le j$. Then $C := AS$ can be computed by Algorithm 2.5.

$\Xi := 0$ from $\mathbb{C}^{I\times p}$
for $i = 1,\dots,\mu$ **do**
  $C_{t_is_i} := A_{t_is_i}S_{s_is_i} + \Xi_{t_i}V_{s_i}^H$
  $\Xi_{t_i} := \Xi_{t_i} + A_{t_is_i}U_{s_i}$
$\Upsilon := 0$ from $\mathbb{C}^{I\times q}$
for $i = \mu,\dots,1$ **do**
  $C_{t_is_i} := C_{t_is_i} + \Upsilon_{t_i}Z_{s_i}^H$
  $\Upsilon_{t_i} := \Upsilon_{t_i} + A_{t_is_i}W_{s_i}$

Algorithm 2.5: $\mathscr{H}$-matrix times semi-separable matrix.

For the computation of SA the blocks have to be ordered with respect to their row indices.

In order to be able to compute AS efficiently, we have to exploit that A is an $\mathscr{H}$-matrix, i.e., that for each block $t \times s \in P$ it holds that $A_{ts} = XY^H$ with $X \in \mathbb{C}^{t\times k}$ and $Y \in \mathbb{C}^{s\times k}$ each consisting of k columns. In this case we have

$$A_{ts}U_s = X(Y^H U_s)$$

and each product $A_{ts}U_s$ appearing in Algorithm 2.5 can be done with $pk(|t|+|s|)$ operations. Additionally, products $A_{ts}S_{ss}$ have to be computed for each block $t \times s \in P$. Exploiting

$$A_{ts}S_{ss} = X(S_{ss}^H Y)^H,$$

it is sufficient to compute the $k\times s$ matrix $S_{ss}^H Y$, which can be done with $\mathscr{O}((p+q)k|s|)$ operations using Algorithm 1.1. The rank-p update of $A_{ts}S_{ss}$ with $\Xi_t V_s^H$ can be done explicitly by storing the rank-$(k+p)$ matrix

$$A_{ts}S_{ss} + \Xi_t V_s^H = [X, \Xi_t][S_{ss}^H Y, V_s]^H,$$

which requires copying $p(|t|+|s|)$ units of storage. The updates with $\Xi_t V_s^H$ and $\Upsilon_t Z_s^H$ lead to a blockwise rank of $k+p+q$, i.e., with $A \in \mathscr{H}(P,k)$ it holds that $AS, SA \in \mathscr{H}(P,k+p+q)$.

We will apply this multiplication to problems (see Sect. 4.6.2) where p and q are constants. Hence, the computational complexity of each block $t \times s$ is of the order $k(|t|+|s|)$, which is exactly the complexity that is required for each block when multiplying an $\mathscr{H}(P,k)$-matrix by a vector. The latter multiplication requires $\mathscr{O}(kn\log n)$ operations; cf. Sect. 2.2.

Remark 2.32. (a) For the update of the LU decomposition we will have to compute the product of triangular hierarchical and triangular semi-separable matrices. In this case the product will be in $\mathscr{H}(P,k+p)$ and $\mathscr{H}(P,k+q)$, respectively, if the hierarchical factor is in $\mathscr{H}(P,k)$. It is obvious how to simplify and accelerate Algorithm 2.5 for such kind of matrices; see Algorithms 2.6 and 2.7.

(b) The rank-p and rank-q updates will gradually increase the rank of the factors if they are stored explicitly. This can be avoided by truncation to rank-k; see Sect. 1.1.4. The latter operation requires $\mathscr{O}(k^2(|t|+|s|))$ operations for each block $t \times s$.

Let $\mathscr{L} = \{t_1 \times s_1, \ldots, t_\mu \times s_\mu\}$ be the blocks in P such that $\min s_i \geq \min s_j$ for $i \leq j$.
$\Upsilon := 0$ from $\mathbb{C}^{I\times q}$
for $i = 1, \ldots, \mu$ **do**
    $C_{t_i s_i} := L_{t_i s_i} L'_{s_i s_i} + \Upsilon_{t_i} Z_{s_i}^H$
    $\Upsilon_{t_i} := \Upsilon_{t_i} + L_{t_i s_i} W_{s_i}$

Algorithm 2.6: $\mathscr{H}$-matrix times lower triangular semi-separable matrix.

Let $\mathscr{L} = \{t_1 \times s_1, \ldots, t_\mu \times s_\mu\}$ be the blocks in P such that $\min t_i \geq \min t_j$ for $i \leq j$.
$\Xi := 0$ from $\mathbb{C}^{I \times p}$
for $i = 1, \ldots, \mu$ **do**
    $C_{t_i s_i} := U'_{t_i t_i} U_{t_i s_i} + U_{t_i} \Xi_{s_i}^H$
    $\Xi_{s_i} := \Xi_{s_i} + U_{t_i s_i}^H V_{t_i}$

Algorithm 2.7: Upper triangular semi-separable matrix times $\mathscr{H}$-matrix.

2.8 Hierarchical Inversion

In this section we assume that each block A_{tt}, $t \in T_I$, of $A \in \mathscr{H}(T_{I\times I}, k)$ is invertible. Positive definite matrices are an important example.

Two approaches to the computation of the $\mathscr{H}$-matrix inverse have been investigated in the literature. The first (see [132]) uses the **Schulz iteration**

$$C_{i+1} = C_i(2I - AC_i), \quad i = 0, 1, 2, \ldots,$$

which arises from Newton's method applied to $F(X) := X^{-1} - A = 0$. This iteration converges locally to A^{-1}; see [235]. The following divide-and-conquer approach (see [127]) has turned out to be significantly more efficient. It exploits the property that each matrix $A \in \mathscr{H}(T_{I\times I}, k)$ is subdivided according to its block cluster tree:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}.$$

It is easy to see that for the exact inverse of A it holds that

$$A^{-1} = \begin{bmatrix} A_{11}^{-1} + A_{11}^{-1} A_{12} S^{-1} A_{21} A_{11}^{-1} & -A_{11}^{-1} A_{12} S^{-1} \\ -S^{-1} A_{21} A_{11}^{-1} & S^{-1} \end{bmatrix}, \tag{2.18}$$

where S denotes the Schur complement $S := A_{22} - A_{21} A_{11}^{-1} A_{12}$ of A_{11} in A. For the computation of A^{-1} the matrices A_{11} and S, which have approximately half the size of A, have to be inverted. The $\mathscr{H}$-matrix inverse C of A is computed by replacing the multiplications and the additions appearing in (2.18) by the $\mathscr{H}$-matrix versions. We need a temporary matrix $T \in \mathscr{H}(T_{I\times I}, k)$, which together with C is initialized to zero.

procedure invertH$(t, A, \mathtt{var}\, C)$
if $t \in \mathscr{L}(T_I)$ **then** $C_{tt} := A_{tt}^{-1}$ is the usual inverse.
else
  let t_1, t_2 denote the sons of t.
  invertH(t_1, A, C).
  $T_{t_1t_2} = T_{t_1t_2} - C_{t_1t_1} A_{t_1t_2}$.
  $T_{t_2t_1} = T_{t_2t_1} - A_{t_2t_1} C_{t_1t_1}$.
  $A_{t_2t_2} = A_{t_2t_2} + A_{t_2t_1} T_{t_1t_2}$.
  invertH(t_2, A, C).
  $C_{t_1t_2} = C_{t_1t_2} + T_{t_1t_2} C_{t_2t_2}$.
  $C_{t_2t_1} = C_{t_2t_1} + C_{t_2t_2} T_{t_2t_1}$.
  $C_{t_1t_1} = C_{t_1t_1} + T_{t_1t_2} C_{t_2t_1}$.

Algorithm 2.8: $\mathscr{H}$-matrix inversion.

Calling invertH with parameters I, A, and C generates the desired approximation C of A^{-1}, while A is destroyed.

The complexity of the computation of the $\mathscr{H}$-inverse is determined by the cost of the $\mathscr{H}$-matrix multiplication. For the following theorem it is assumed that each sum of two rank-k matrices arising during the previous algorithm is rounded to rank k. We remark that the feasibility of this assumption has to be checked for the respective problem since otherwise the approximation error may become uncontrollable.

Theorem 2.33. *The computation of the $\mathscr{H}$-matrix inverse $C \in \mathscr{H}(T_{I\times I}, k)$ of $A \in \mathscr{H}(T_{I\times I}, k)$ using Algorithm 2.8 requires $\mathscr{O}(k^2 L^2(T_I)|I|)$ operations.*

If we prescribe the rounding precision $\varepsilon_{\mathscr{H}}$, it is by no means obvious that the required blockwise rank k of the $\mathscr{H}$-matrix inverse C is small. In order to prove this, we will leave our algebraic point of view and exploit analytic properties which are accessible for subclasses of matrices such as those arising from elliptic operators; see Chap. 4.

The $\mathscr{H}$-matrix inverse may be used for the data-sparse approximation of operator valued functions. Let $f : \mathbb{C} \to \mathbb{C}$ be analytic inside of a path $\Gamma \subset \mathbb{C}$ enveloping the spectrum of an elliptic operator $\mathscr{L}$. The operator $f(\mathscr{L})$ can be represented using the **Dunford-Cauchy integral**

$$f(\mathscr{L}) = \frac{1}{2\pi \mathrm{i}} \int_\Gamma f(z)(zI - \mathscr{L})^{-1}\, \mathrm{d}z$$

provided that this integral converges. Approximating the previous integral by appropriate quadrature formula and treating the discrete resolvents by the $\mathscr{H}$-matrix inverse leads to a data-sparse approximation. Examples are the **operator exponential** $\exp(-t\mathscr{L})$ arising from the solution of the heat equation and the **operator cosine family** $\cos(t\sqrt{\mathscr{L}})$ arising from the wave equation; see [98, 96]. In addition, data-sparse methods for the approximation of the Sylvester and the Riccati equation including the matrix sign-function are investigated in [96, 97].

2.8.0.1 Updates of the Inverse

Sometimes, many systems with coefficient matrices differing in only a small number of entries have to be solved. This problem arises, for instance, as a consequence of Newton's method for the solution of nonlinear systems if the coefficients of the operator change only locally. Assume that A is invertible such that $1+\alpha e_j^T A^{-1} e_i \neq 0$ with $\alpha \in \mathbb{R}$ and the canonical vectors $e_i, e_j \in \mathbb{R}^n$. Due to the Sherman-Morrison-Woodbury formula (1.3) for the inverse of

$$\tilde{A} := A + \alpha e_i e_j^T$$

it holds that

$$\tilde{A}^{-1} = (A + \alpha e_i e_j^T)^{-1} = A^{-1} - \frac{\alpha}{1+\alpha e_j^T A^{-1} e_i} A^{-1} e_i e_j^T A^{-1}.$$

Hence, $\tilde{A}^{-1}$ and A^{-1} differ only by matrix of rank 1. In the case that A and $\tilde{A}$ have r different entries, the update will be of rank at most r. Using the $\mathscr{H}$-matrix addition, we are able to compute an approximation $C \in \mathscr{H}(T_{I\times I}, k)$ of $\tilde{A}^{-1}$ exploiting the previously computed approximation of A^{-1}. Obviously, this is much faster than computing the $\mathscr{H}$-matrix inverse of $\tilde{A}$ from scratch.

2.9 Computing the $\mathscr{H}$-Matrix LU Decomposition

Although the hierarchical inversion has almost linear complexity provided that the required blockwise rank behaves well, the numerical effort for its computation is still relatively high. The following hierarchical LU decomposition provides a significantly more efficient alternative.

The first algorithm for the computation of hierarchical LU decompositions has been proposed in [176]. This algorithm, however, was defined on a partition which is too restrictive to treat problems of higher spatial dimension. The following algorithm is the first method (see [21]) which can be applied to general $\mathscr{H}$-matrices. Once again, the required blockwise rank cannot be predicted without restricting the class of matrices. For elliptic operators it will be possible to prove a logarithmic dependence on $|I|$; cf. Sect. 4.3.

We have seen that approximate versions of the usual matrix operations like addition and multiplication can be defined on the set $\mathscr{H}(T_{I\times I}, k)$ of hierarchical matrices. The hierarchical LU decomposition can be computed using these accelerated operations during the block LU decomposition instead of the usual ones. The rounding precision these operations are performed with will be denoted by $\varepsilon_{\mathscr{H}}$.

To define the $\mathscr{H}$-matrix LU decomposition, we exploit the hierarchical block structure of a block A_{tt}, $t \in T_I \setminus \mathscr{L}(T_I)$:

$$A_{tt} = \begin{bmatrix} A_{t_1t_1} & A_{t_1t_2} \\ A_{t_2t_1} & A_{t_2t_2} \end{bmatrix} = \begin{bmatrix} L_{t_1t_1} & \\ L_{t_2t_1} & L_{t_2t_2} \end{bmatrix} \begin{bmatrix} U_{t_1t_1} & U_{t_1t_2} \\ & U_{t_2t_2} \end{bmatrix},$$

where $t_1, t_2 \in T_I$ denote the sons of t in T_I. Hence, the LU decomposition of a block A_{tt} is reduced to the following four problems on the sons of $t \times t$:

(i) Compute $L_{t_1t_1}$ and $U_{t_1t_1}$ from the LU decomposition $L_{t_1t_1}U_{t_1t_1} = A_{t_1t_1}$;
(ii) Compute $U_{t_1t_2}$ from $L_{t_1t_1}U_{t_1t_2} = A_{t_1t_2}$;
(iii) Compute $L_{t_2t_1}$ from $L_{t_2t_1}U_{t_1t_1} = A_{t_2t_1}$;
(iv) Compute $L_{t_2t_2}$ and $U_{t_2t_2}$ from the LU decomposition $L_{t_2t_2}U_{t_2t_2} = A_{t_2t_2} - L_{t_2t_1}U_{t_1t_2}$.

If a block $t \times t \in \mathscr{L}(T_{I\times I})$ is a leaf, the usual pivoted LU decomposition is employed. For (i) and (iv) two LU decompositions of half the size have to be computed. In order to solve (ii), i.e., solve a problem of the structure $L_{tt}B_{ts} = A_{ts}$ for B_{ts}, where L_{tt} is a lower triangular matrix and $t \times s \in T_{I\times I}$, we use the following recursive block forward substitution. If the block $t \times s$ is not a leaf in $T_{I\times I}$, from the subdivision of the blocks A_{ts}, B_{ts}, and L_{tt} into their sub-blocks (t_1, t_2 and s_1, s_2 are again the sons of t and s, respectively)

$$\begin{bmatrix} L_{t_1t_1} & \\ L_{t_2t_1} & L_{t_2t_2} \end{bmatrix} \begin{bmatrix} B_{t_1s_1} & B_{t_1s_2} \\ B_{t_2s_1} & B_{t_2s_2} \end{bmatrix} = \begin{bmatrix} A_{t_1s_1} & A_{t_1s_2} \\ A_{t_2s_1} & A_{t_2s_2} \end{bmatrix}$$

one observes that B_{ts} can be found from the following equations

$$\begin{aligned} L_{t_1t_1}B_{t_1s_1} &= A_{t_1s_1}, \\ L_{t_1t_1}B_{t_1s_2} &= A_{t_1s_2}, \\ L_{t_2t_2}B_{t_2s_1} &= A_{t_2s_1} - L_{t_2t_1}B_{t_1s_1}, \\ L_{t_2t_2}B_{t_2s_2} &= A_{t_2s_2} - L_{t_2t_1}B_{t_1s_2}, \end{aligned}$$

which are again of type (ii). If, on the other hand, $t \times s$ is a leaf, then the usual forward substitution is applied. Similarly, one can solve (iii) by recursive block backward substitution.

The complexity of the above recursions is determined by the complexity of the hierarchical matrix multiplication, which can be estimated as $\mathscr{O}(k^2L^2(T_I)|I|)$ for two matrices from $\mathscr{H}(T_{I\times I}, k)$. Each operation is carried out with precision $\varepsilon_{\mathscr{H}}$. A result [79] on the stability analysis of the block LU decomposition states that the product LU is backward stable in the sense that

$$\|A - LU\|_2 < c(|I|)\varepsilon_{\mathscr{H}}(\|A\|_2 + \|L\|_2\|U\|_2).$$

Provided that $\|L\|_2\|U\|_2 \approx \|A\|_2$, the relative accuracy of LU will hence be of order $\varepsilon_{\mathscr{H}}$. Employing the $\mathscr{H}$-matrix arithmetic, it is therefore possible to generate an approximate LU decomposition of a matrix $A \in \mathscr{H}(T_{I\times I}, k)$ to any prescribed accuracy with almost linear complexity whenever the blockwise rank is guaranteed to be logarithmically bounded. A logarithmic dependence of k on $|I|$ will, for instance, be proved under quite general assumptions for finite element stiffness matrices arising from the discretization of elliptic boundary value problems.

It is known that the pointwise LU decomposition preserves the bandwidth and the skyline structure of a matrix. This property is obviously inherited by the $\mathscr{H}$-matrix LU decomposition. In Sect. 4.5 we will exploit this in the context of nested dissection reorderings, which in particular allows to parallelize the LU factorization algorithm.

In the case of positive definite matrices A it is possible to define an $\mathscr{H}$-matrix version of the Cholesky decomposition of a block A_{tt}, $t \in T_I \setminus \mathscr{L}(T_I)$:

$$A_{tt} = \begin{bmatrix} A_{t_1t_1} & A_{t_1t_2} \\ A_{t_1t_2}^H & A_{t_2t_2} \end{bmatrix} = \begin{bmatrix} L_{t_1t_1} & \\ L_{t_2t_1} & L_{t_2t_2} \end{bmatrix} \begin{bmatrix} L_{t_1t_1} & \\ L_{t_2t_1} & L_{t_2t_2} \end{bmatrix}^H.$$

This factorization is recursively computed by

$$\begin{aligned} L_{t_1t_1} L_{t_1t_1}^H &= A_{t_1t_1}, \\ L_{t_1t_1} L_{t_2t_1}^H &= A_{t_1t_2}, \\ L_{t_2t_2} L_{t_2t_2}^H &= A_{t_2t_2} - L_{t_2t_1} L_{t_2t_1}^H \end{aligned}$$

using the usual Cholesky decomposition on the leaves of $T_{I\times I}$. The second equation $L_{t_1t_1} L_{t_2t_1}^H = A_{t_1t_2}$ is solved for $L_{t_2t_1}$ in a way similar to how $U_{t_1t_2}$ was previously obtained in the LU decomposition.

Once $A \approx L_{\mathscr{H}} U_{\mathscr{H}}$ has been decomposed, the solution of $Ax = b$ can be found by forward/backward substitution: $L_{\mathscr{H}} y = b$ and $U_{\mathscr{H}} x = y$. An advantage of the inverse of a matrix A is that A^{-1} only has to be multiplied by b when solving the linear system $Ax = b$ for x. The LU decomposition requires forward-/backward substitution, which in standard arithmetic has quadratic complexity. Since $L_{\mathscr{H}}$ and $U_{\mathscr{H}}$ are $\mathscr{H}$-matrices, y_t, $t \in T_I \setminus \mathscr{L}(T_I)$, can be computed recursively by solving the following systems for y_{t_1} and y_{t_2}

$$L_{t_1t_1} y_{t_1} = b_{t_1} \quad \text{and} \quad L_{t_2t_2} y_{t_2} = b_{t_2} - L_{t_2t_1} y_{t_1}.$$

If $t \in \mathscr{L}(T_I)$ is a leaf, a usual triangular solver is used. The backward substitution can be done analogously. These substitutions are exact and their complexity is determined by the complexity of the hierarchical matrix-vector multiplication, which is $\mathscr{O}(kL(T_I)|I|)$ for multiplying an $\mathscr{H}(T_{I\times I}, k)$-matrix by a vector.

2.10 Hierarchical QR Decomposition

In addition to the hierarchical LU decomposition it seems straight forward to define a hierarchical QR decomposition of $A \in \mathscr{H}(T_{I\times I}, k)$, which may, for instance, be used to solve eigenvalue problems with almost linear complexity. In [176] the QR decomposition of $\mathscr{H}$-matrices is computed using the Cholesky decomposition $LL^H = B$ of $B := A^H A$. The matrix Q is found by forward substitution from $A = QL^H$. Then Q is unitary because

$$Q^H Q = L^{-1} A^H A L^{-H} = L^{-1} B L^{-H} = I.$$

Since squaring a matrix should be avoided, we propose to use the following alternative recursion.

Assume that A or a diagonal sub-block of A is partitioned in the following way:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}.$$

Setting $X := A_{21} A_{11}^{-1}$, it is obvious that $I + X^H X$ and $I + X X^H$ are Hermitian positive definite and share the same spectra (without 1). Let the Cholesky factors L_1 and L_2 be computed from

$$I + X^H X = L_1 L_1^H \quad \text{and} \quad I + X X^H = L_2 L_2^H,$$

then it follows that $\det L_1 = \det L_2$. The matrix

$$Q := \begin{bmatrix} L_1^{-1} & L_1^{-1} X^H \\ -L_2^{-1} X & L_2^{-1} \end{bmatrix} = \begin{bmatrix} L_1^{-1} & \\ & L_2^{-1} \end{bmatrix} \begin{bmatrix} I & X^H \\ -X & I \end{bmatrix}$$

is unitary and satisfies

$$\begin{aligned} QA &= \begin{bmatrix} L_1^{-1}(A_{11} + X^H A_{21}) & L_1^{-1}(A_{12} + X^H A_{22}) \\ 0 & L_2^{-1}(A_{22} - X A_{12}) \end{bmatrix} \\ &= \begin{bmatrix} L_1^H A_{11} & L_1^{-1}(A_{12} + X^H A_{22}) \\ 0 & L_2^{-1} S \end{bmatrix}, \end{aligned} \tag{2.19}$$

where $S := A_{22} - X A_{12}$ is the Schur complement of A_{11} in A. If one of the diagonal blocks of QA is a leaf in the block cluster tree, then the usual QR decomposition is applied to it. Otherwise, two appropriate transformations of half the size have to be computed and applied. Instead of immediately applying the next transformation to the upper right block of (2.19), it is more efficient to gradually apply the transformations to each other before they are multiplied by this block. In this case, the whole QR recursion is determined by the complexity of the $\mathscr{H}$-matrix multiplication, which is of order $k^2 L^2(T_I)|I|$.

The matrix Q can be regarded as a block Givens rotation since $\det Q = 1$, which follows from

$$\det Q = \det \begin{bmatrix} L_1^{-1} & L_1^{-1} X^H \\ 0 & L_2^{-1}(I + X X^H) \end{bmatrix} = \det \begin{bmatrix} L_1^{-1} & L_1^{-1} X^H \\ 0 & L_2^H \end{bmatrix}.$$

Since the computation of the factors L_1 and L_2 involves approximation errors, Q will be only "approximately" unitary; i.e., there is an error matrix $E \in \mathbb{C}^{I \times I}$ such that

$$(Q + E)^H (Q + E) = I.$$

Since the loss of orthogonality of the columns of Q is amplified by a bad condition number of A, Q can be improved by decomposing $Q = Q'R'$, which leads to $A = Q'\hat{R}$ with the upper block triangular matrix $\hat{R} := R'R$.

In order to transform A to an upper triangular matrix, we have to apply a sequence of approximately unitary matrices $Q_1, \ldots, Q_L$, where $L := L(T_I) \sim \log |I|$ denotes the depth of T_I. The following lemma states that the distance of the product $Q_1 \cdot \ldots \cdot Q_L$ to a unitary matrix is of the order $\varepsilon \log L$ if the rounding precision is increased from level to level as ε/ℓ.

Lemma 2.34. *Let $0 < \varepsilon < 1$. Assume that Q_ℓ, $\ell = 1, \ldots, L$, are approximately unitary in the sense that $(Q_\ell + E_\ell)^H (Q_\ell + E_\ell) = I$ for some matrices $E_\ell \in \mathbb{C}^{I \times I}$ satisfying $\|E_\ell\|_2 < \varepsilon/\ell$. Then also the product $Q_L \cdot \ldots \cdot Q_1$ is approximately unitary; i.e.,*

$$(Q_L \cdot \ldots \cdot Q_1 + E)^H (Q_L \cdot \ldots \cdot Q_1 + E) = I$$

for some $E \in \mathbb{C}^{I \times I}$ satisfying $\|E\|_2 \sim \varepsilon \log L$.

Proof. The assertion is proved by induction over L. The case $L = 1$ is trivially true. Assume that for $Q := Q_L \cdot \ldots \cdot Q_1$ it holds that

$$(Q+E)^H (Q+E) = I$$

for some $E \in \mathbb{C}^{I \times I}$ satisfying $\|E\|_2 \le \varepsilon \left(\sum_{\ell=1}^{L} 1/\ell\right) \prod_{\ell=2}^{L} (1 + \frac{\varepsilon}{\ell})$. Observe that

$$[Q_{L+1}Q+F]^H [Q_{L+1}Q+F] = [(Q_{L+1}+E_{L+1})(Q+E)]^H (Q_{L+1}+E_{L+1})(Q+E) = I,$$

where $F := (Q_{L+1} + E_{L+1})E + E_{L+1}(Q+E) - E_{L+1}E$. The norm of F can be estimated as

$$\begin{aligned}
\|F\|_2 &\le \|E\|_2 + \|E_{L+1}\|_2 + \|E_{L+1}\|_2 \|E\|_2 \\
&\le \varepsilon \left(\sum_{\ell=1}^{L} 1/\ell \right) \prod_{\ell=2}^{L} (1 + \frac{\varepsilon}{\ell}) + \frac{\varepsilon}{L+1} + \frac{\varepsilon^2}{L+1} \left(\sum_{\ell=1}^{L} 1/\ell \right) \prod_{\ell=2}^{L} (1 + \frac{\varepsilon}{\ell}) \\
&\le \varepsilon \left(\sum_{\ell=1}^{L+1} 1/\ell \right) \prod_{\ell=2}^{L} (1 + \frac{\varepsilon}{\ell}) + \frac{\varepsilon^2}{L+1} \left(\sum_{\ell=1}^{L+1} 1/\ell \right) \prod_{\ell=2}^{L} (1 + \frac{\varepsilon}{\ell}) \\
&= \varepsilon \left(\sum_{\ell=1}^{L+1} 1/\ell \right) \prod_{\ell=2}^{L+1} (1 + \frac{\varepsilon}{\ell}).
\end{aligned}$$

The assertion follows from $\prod_{\ell=2}^{L} (1 + \frac{\varepsilon}{\ell}) \le \exp(\varepsilon)$ and $\sum_{\ell=1}^{L} 1/\ell \sim \log L$. □

Assume a logarithmic dependence of the required rank k on the prescribed accuracy. Then increasing the accuracy with the level ℓ does not significantly increase the rank due to $k \sim |\log \varepsilon/\ell| \sim |\log \varepsilon| + \log \ell \sim |\log \varepsilon| + \log\log |I|$.

2.11 $\mathcal{H}^2$-Matrices and Fast Multipole Methods

The low-rank blocks of $\mathcal{H}$-matrices can be chosen independently from each other. By fixing a common basis for the column and the row vectors it is possible to further reduce the complexity of data-sparse approximations.

Definition 2.35. Let T_I be a cluster tree and let $k_t \in \mathbb{N}$, $t \in T_I$, be given. A family of matrices $V(t) \in \mathbb{C}^{t\times k_t}$ is called **cluster basis** for the cluster tree T_I and the rank distribution $(k_t)_{t\in T_I}$.

Let cluster bases $\mathcal{U} := \{U(t), t \in T_I\}$ and $\mathcal{V} := \{V(s), s \in T_J\}$ with associated rank distributions $(k_t^{\mathcal{U}})_{t\in T_I}$ and $(k_s^{\mathcal{V}})_{s\in T_J}$ be given. The following set of uniform $\mathcal{H}$-matrices (see [127]) is a subset of the set of $\mathcal{H}$-matrices.

Definition 2.36. Let $T_{I\times J}$ be a block cluster tree generated from the cluster trees T_I and T_J. Furthermore, let $\mathcal{U}$ and $\mathcal{V}$ be cluster bases for T_I and T_J. A matrix $A \in \mathbb{C}^{I\times J}$ satisfying

$$A_{ts} = U(t)S(t,s)V(s)^H \quad \text{for all admissible } t\times s \in \mathcal{L}(T_{I\times J}) \tag{2.20}$$

and some $S(t,s) \in \mathbb{C}^{k_t^{\mathcal{U}}\times k_s^{\mathcal{V}}}$ is called **uniform hierarchical matrix** for $\mathcal{U}$ and $\mathcal{V}$.

Condition (2.20) means that A_{ts} is in the subspace

$$\operatorname{span}\{U_i(t)V_j(t)^H,\ i = 1,\dots,k_t^{\mathcal{U}},\ j = 1,\dots,k_s^{\mathcal{V}}\},$$

where U_i and V_j denote the ith and the jth column of U and V, respectively. In contrast to $\mathcal{H}$-matrices, uniform $\mathcal{H}$-matrices of *common* cluster bases form a linear subspace of $\mathbb{C}^{I\times J}$, which avoids rounding sums. Note that this subspace property would also hold under the weaker condition that each of the two cluster bases depends on *t and s*, i.e.,

$$A_{ts} = U(t,s)S(t,s)V(t,s)^H$$

instead of (2.20).

Let $k = \max\{k_t^{\mathcal{U}}, k_s^{\mathcal{V}}; t \in T_I, s \in T_J\}$. If many uniform $\mathcal{H}$-matrices of common cluster bases $\mathcal{U}$ and $\mathcal{V}$ are to be stored, $\mathcal{U}$ and $\mathcal{V}$ should be stored separately from the coefficient matrices $S(t,s)$. The storage requirements of the latter are of the order $k\min\{|I|,|J|\}$, which due to

$$\sum_{t\in T_I}\sum_{t\times s\in P} \min\{k^2,|t|^2\} \le c_{\mathrm{sp}} \sum_{t\in T_I} \min\{k^2,|t|^2\}$$

can be seen from (1.15). Here, we have assumed that for clusters t,s from the same level it holds that $|t| \approx |s|$.

However, storing the cluster bases still requires $k[|I|L(T_I) + |J|L(T_J)]$ units of storage due to Lemma 1.21. This situation can be improved by introducing a second hierarchy. The following space of $\mathcal{H}^2$-matrices (see [136]) consists of uniform $\mathcal{H}$-matrices of given cluster bases $\mathcal{U}$ and $\mathcal{V}$ which are nested.

Definition 2.37. A cluster basis $\mathscr{V}$ is called **nested** if for each $t \in T \setminus \mathscr{L}(T)$ there are transfer matrices $B_{t't} \in \mathbb{C}^{k_{t'} \times k_t}$ such that

$$(V(t))_{t'} = V(t')B_{t't} \quad \text{for all } t' \in S(t).$$

Storing a nested cluster basis requires storing $V(t)$ for all leaf clusters $t \in \mathscr{L}(T)$ and the transfer matrices $B_{t't}$, $t' \in S(t)$, for all $t \in T \setminus \mathscr{L}(T_I)$, which can be done with order $k|I|$ units of storage due to (1.15); cf. [129].

Definition 2.38. An **$\mathscr{H}^2$-matrix** is a uniform $\mathscr{H}$-matrix with nested cluster bases.

While in [136] the Taylor expansion is used to show existence of $\mathscr{H}^2$-matrix approximants, in [103] a practical procedure for their construction is proposed which is based on polynomial interpolation. Since polynomials do not provide an optimal approximation system, algebraic recompression techniques [44, 41] can be used to improve the approximant. The complexity $k|I|$ can also be achieved by the method presented in Sect. 3.5, which uses the conceptionally easier uniform $\mathscr{H}$-matrices.

2.11.0.2 Matrix-Vector Multiplication

The matrix-vector multiplication $y := y + Ax$ of an $\mathscr{H}^2$-matrix A by a vector $x \in \mathbb{C}^J$ can be done by the following three-phase algorithm; cf. [136].

1. *Forward transform*
   In this first phase, transformed vectors $\hat{x}(s) := V(s)^H x_s$ are computed for all $s \in T_J$. Exploiting the nestedness of the cluster basis $\mathscr{V}$, one has the following recursive relation

   $$\hat{x}(s) = V(s)^H x_s = \sum_{s' \in S(s)} B_{s's}^H V(s')^H x_{s'} = \sum_{s' \in S(s)} B_{s's}^H \hat{x}(s'),$$

   which has to be applied starting from the leaf vectors $\hat{x}(s) = V(s)^H x_s$, $s \in \mathscr{L}(T_J)$.
2. *Far field interaction*
   In the second phase the products $S(t,s)\hat{x}(s)$ are computed and summed up over all clusters in $R_t := \{s \in T_J : t \times s \in P \text{ is admissible}\}$:

   $$\hat{y}(t) := \sum_{s \in R_t} S(t,s)\hat{x}(s), \quad t \in T_I.$$

3. *Backward transform*
   In this third phase, the vectors $\hat{y}(t)$ are transformed to the target vector y. The nestedness of the cluster basis $\mathscr{U}$ provides the following recursion which descends the cluster tree T_I

   (a) Compute $\hat{y}(t') := \hat{y}(t') + B_{t't}\hat{y}(t)$ for all $t' \in S(t)$;
   (b) Compute $y_t := y_t + U(t)\hat{y}(t)$ for all leaf clusters $t \in \mathscr{L}(T_I)$.

4. *Near field interaction*
   Compute $y_t := y_t + A_{ts} x_s$ for all non-admissible blocks $t \times s \in P$.

The previous matrix-vector multiplication is an algebraic generalization of the **fast multipole method**; see [121] and the improved version [122]. The number of operations is of the order of the number of involved matrix entries and hence $k[|I| + |J|]$. $\mathscr{H}^2$-matrices will therefore improve the asymptotic complexity of $\mathscr{H}$-matrices by a single logarithmic factor due to the nestedness of the cluster bases. If **variable order techniques** are used, i.e., the rank

$$k(\ell) = [\alpha(\mathscr{L}(T_I) - \ell) + \beta]^\gamma$$

with parameters α, β, and γ is chosen depending on the level ℓ of a block, then one can guarantee linear complexity; cf. [136, 227, 184]. Variable order approximations, however, are feasible only in very special situations.

In addition to the matrix-vector multiplication, $\mathscr{H}^2$-matrices admit matrix operations such as matrix addition and matrix multiplication with different cluster bases; see [42] for addition and multiplication algorithms of complexity $k^3|I|$. For these operations, however, the problem arises that the cluster bases required to hold the results of addition and multiplication differ from the bases of the input matrices and are usually unknown.

2.12 Using Hierarchical Matrices for Preconditioning

When investigating the complexity of algorithms for the solution of linear systems arising from the discretization of a continuous problem, one considers a sequence of systems

$$A_n x_n = b_n, \quad n \to \infty, \tag{2.21}$$

where each $A_n \in \mathbb{C}^{n \times n}$ is invertible. However, for the sake of readability the index n is usually dropped whenever this dependency is obvious from the context. For the solution of (2.21) either direct or iterative solvers can be used. The complexity of direct solvers such as Gaussian elimination applied to fully populated linear systems is of cubic order and hence prohibitively large. There are improved variants (cf. [78, 4, 230]) of Gaussian elimination if A is sparse. Due to *fill-in*, these methods are efficient only in two spatial dimensions. The presented hierarchical LU decomposition from Sect. 2.9 could in principle be used to compute an approximate LU decomposition with almost linear complexity. $\mathscr{H}$-matrices, however, can be multiplied by a vector with complexity $kn \log n$ and a much smaller (hidden) constant. The iterative Krylov subspace methods (cf. [223]) such as the **conjugate gradient method** (CG) [145] or the **method of generalized minimal residuals** (GMRes) [224], which the matrix A enters only through the matrix-vector product, will hence be faster provided that the number of iterations is small enough.

It is well known that the convergence of Krylov subspace methods is determined by spectral properties such as the distribution of eigenvalues if A is normal or the numerical range in the general case. Depending on the mapping properties of the underlying differential or integral operator $\mathscr{A}$, its discretization A and hence also its $\mathscr{H}$-matrix approximant $A_{\mathscr{H}}$ may be ill-conditioned. In addition, a large condition number can result from the coefficients of the operator or the discretization of the geometry even for small n; see Sect. 3.6. Therefore, if (2.21) is to be solved iteratively, one has to incorporate a preconditioner.

A **left preconditioner** is a regular and easily invertible matrix C such that $C \approx A_{\mathscr{H}}^{-1}$ in the sense that the distribution of eigenvalues of $CA_{\mathscr{H}}$ leads to an improved convergence of the respective iteration scheme, where instead of (2.21) one solves the equivalent linear system

$$CA_{\mathscr{H}}x = Cb.$$

The convergence rate of Krylov subspace methods in the Hermitian case is determined by the spectral condition number of $CA_{\mathscr{H}}$. Hence, C has to be chosen such that $\mathrm{cond}_2(CA_{\mathscr{H}}) \sim 1$. If $A_{\mathscr{H}}$ is non-Hermitian, the aim of preconditioning is to obtain a spectrum of $CA_{\mathscr{H}}$ which is clustered away from the origin. When Krylov subspace methods are used, it is not necessary to form the preconditioned matrix $CA_{\mathscr{H}}$. Instead, vectors should be multiplied by $A_{\mathscr{H}}$ and C consecutively. If C is used as a **right preconditioner**, (2.21) is replaced by

$$A_{\mathscr{H}}C\tilde{x} = b.$$

In the latter case, the solution x can be obtained as $x = C\tilde{x}$. In this section only right preconditioners are considered. Left preconditioners can be constructed analogously.

Since $\mathscr{H}$-matrices provide efficient approximations to the inverse and to the LU decomposition, they are particularly suited for preconditioning. Although the results of this section are proved for the approximate inverse, they are obviously also valid if the approximate LU decomposition $A_{\mathscr{H}} \approx L_{\mathscr{H}}U_{\mathscr{H}}$ is used as $C = (L_{\mathscr{H}}U_{\mathscr{H}})^{-1}$. The aim of this section is to establish a relation between the condition number of $A_{\mathscr{H}}C$ and the approximation accuracy ε of the inverse in the Hermitian case. For non-Hermitian coefficient matrices we will present lower bounds for the distance of a cluster of eigenvalues of $A_{\mathscr{H}}C$ and the origin; cf. [22]. This relation will be used to find out the required size of ε. It will be seen that a low-accuracy approximate inverse of $A_{\mathscr{H}}$ is sufficient to guarantee a bounded number of preconditioned iterations of appropriate iterative schemes. In addition, the derived condition will guarantee required properties of the preconditioner such as invertibility and positivity. Moreover, the number of iterations will depend neither on the operator nor on the computational domain but only on the accuracy ε. Numerical experiments in Sect. 3.6 and Sect. 4.4 will demonstrate the effectiveness of the preconditioner when it is applied to fully populated matrices arising from the discretization of integral operators and to sparse discretizations of partial differential operators.

In the first part of this section the case of Hermitian positive definite coefficient matrices $A_{\mathscr{H}}$ is treated, the second part concentrates on the non-Hermitian case.

2.12.1 Hermitian Positive Definite Coefficient Matrices

Depending on the operator $\mathscr{A}$, the approximation $A_{\mathscr{H}}$ to the discrete operator $A \in \mathbb{R}^{n\times n}$ is often Hermitian positive definite. If (2.21) is to be solved iteratively, the **preconditioned conjugate gradient method** (PCG) is the method of choice. Its convergence rate and hence the number of iterations required to obtain a prescribed accuracy of the solution is determined by the distribution of eigenvalues of $A_{\mathscr{H}}$; see Theorem 2.40.

Typically, the condition number of A grows for an increasing number of unknowns n. The aim of this section is to present preconditioners C such that the number of iterations for the preconditioned coefficient matrix $A_{\mathscr{H}}C$ is bounded by a constant. A bounded number of iterations, in turn, is guaranteed by an asymptotically well-conditioned matrix $A_{\mathscr{H}}C$.

Definition 2.39. Let $\{A_n\}_{n\in\mathbb{N}}$ be a sequence of Hermitian matrices. Assume that there is a constant $c > 0$ such that

$$\operatorname{cond}_2(A_n) \le c \tag{2.22}$$

for all $n \in \mathbb{N}$. Then $\{A_n\}_{n\in\mathbb{N}}$ is said to be **asymptotically well-conditioned**.

The following theorem (cf. [9]) describes the convergence of the conjugate gradient method. The estimate is formulated in terms of the norm $\|x\|_{A_{\mathscr{H}}} := \|A_{\mathscr{H}}x\|_2$.

Theorem 2.40. *Let the spectrum of $A_{\mathscr{H}}C$ be decomposed in the following way:*

$$\sigma(A_{\mathscr{H}}C) = \{\lambda_i', i = 1,\dots,p\} \cup \Lambda \cup \{\lambda_i'', i = 1,\dots,q\}, \quad \Lambda \subset [a,b].$$

Then for the error $x_k - x$, $k = 0,\dots,n-1$, of the iterate x_k of PCG it holds that

$$\|x_k - x\|_{A_{\mathscr{H}}} \le 2(\operatorname{cond}_2(A_{\mathscr{H}}C)+1)^p \left(\frac{\sqrt{b/a}-1}{\sqrt{b/a}+1}\right)^{k-p-q} \|x_0 - x\|_{A_{\mathscr{H}}}.$$

Hence, if $A_{\mathscr{H}}C$ is asymptotically well-conditioned, we may choose $\Lambda = \sigma(A_{\mathscr{H}}C)$. In this case, PCG converges linearly, and the number of iterations depends only on the condition number of $A_{\mathscr{H}}C$ and not on n.

Although asymptotically well-conditioned coefficient matrices lead to a bounded number of iterations, this number might still be large since the constant in (2.22) usually depends on the coefficients of the underlying operator $\mathscr{A}$, the geometry, and its discretization. This influence might be even more severe than the dependence on n; see the numerical experiments in Sect. 3.6 and Sect. 4.4. As we shall see in the next lemma, the condition

$$\|I - A_{\mathscr{H}}C\|_2 \le \varepsilon < 1, \tag{2.23}$$

in which ε does not depend on n, leads to an asymptotically well-conditioned matrix $A_{\mathscr{H}}C$. Spectral equivalence of two matrices A and B does not require A to

approximate B. The matrix $2B$, for instance, has the same preconditioning effect as B. However, if they approximate in a certain sense, any condition number in the neighborhood of 1 can be achieved by decreasing the approximation error. Especially, this allows to guarantee problem-independent convergence rates, whereas non-approximating preconditioners usually are only able to guarantee the boundedness of the condition number.

Lemma 2.41. *Assume that* (2.23) *holds. Then*

$$\operatorname{cond}_2(A_{\mathscr{H}}C) = \|A_{\mathscr{H}}C\|_2\|(A_{\mathscr{H}}C)^{-1}\|_2 \leq \frac{1+\varepsilon}{1-\varepsilon}. \tag{2.24}$$

Proof. The assertion follows from the triangle inequality

$$\|A_{\mathscr{H}}C\|_2 \leq \|I\|_2 + \|I - A_{\mathscr{H}}C\|_2 \leq 1+\varepsilon$$

and from the Neumann series

$$\|(A_{\mathscr{H}}C)^{-1}\|_2 \leq \sum_{k=0}^{\infty} \|I - A_{\mathscr{H}}C\|_2^k = \frac{1}{1-\varepsilon}.$$

□

Note that (2.24) provides an explicit bound on the condition number. The choice $\varepsilon = 0.5$, for instance, guarantees that $\operatorname{cond}_2(A_{\mathscr{H}}C) \leq 3$. In addition, condition (2.23) guarantees that C is non-singular. To see this, let λ be an eigenvalue of $A_{\mathscr{H}}C$ with associated eigenvector z, $\|z\|_2 = 1$, then

$$|1-\lambda| = \|(I - A_{\mathscr{H}}C)z\|_2 \leq \|I - A_{\mathscr{H}}C\|_2 \leq \varepsilon < 1. \tag{2.25}$$

Hence, with $A_{\mathscr{H}}C$ also C is non-singular. In order to be able to apply PCG, C additionally needs to be Hermitian positive definite. It is interesting to see that this is already guaranteed by condition (2.23).

Lemma 2.42. *Assume that $A_{\mathscr{H}}$ is Hermitian positive definite. Then any Hermitian matrix C satisfying* (2.23) *is positive definite, too.*

Proof. According to the assumptions, the square root $A_{\mathscr{H}}^{1/2}$ of $A_{\mathscr{H}}$ is defined. Since $A_{\mathscr{H}}C$ is similar to the Hermitian matrix $A_{\mathscr{H}}^{1/2}CA_{\mathscr{H}}^{1/2}$, the eigenvalues of $A_{\mathscr{H}}C$ are real. Moreover, for the smallest eigenvalue of $A_{\mathscr{H}}^{1/2}CA_{\mathscr{H}}^{1/2}$ it follows from (2.25) that

$$\lambda_{\min}(A_{\mathscr{H}}^{1/2}CA_{\mathscr{H}}^{1/2}) = \lambda_{\min}(A_{\mathscr{H}}C) \geq 1-\varepsilon.$$

Let $x \neq 0$ and $y = A_{\mathscr{H}}^{-1/2}x$. Then $y \neq 0$ and we have

$$x^H Cx = y^H A_{\mathscr{H}}^{1/2}CA_{\mathscr{H}}^{1/2}y \geq (1-\varepsilon)\|y\|_2^2 > 0,$$

which proves that C is positive definite. □

From the approximation by $\mathcal{H}$-matrices usually error estimates of the form

$$\|A_{\mathcal{H}} - C^{-1}\|_2 \le \varepsilon \|A_{\mathcal{H}}\|_2 \quad \text{or} \quad \|A_{\mathcal{H}}^{-1} - C\|_2 \le \varepsilon \|A_{\mathcal{H}}^{-1}\|_2 \tag{2.26}$$

instead of (2.23) are satisfied. In this case, the following lemma describes a sufficient condition on ε.

Lemma 2.43. *Assume that* (2.26) *holds with* $\varepsilon > 0$ *such that* $\varepsilon' := \varepsilon \operatorname{cond}_2(A_{\mathcal{H}}) < 1$. *Then*

$$\operatorname{cond}_2(A_{\mathcal{H}} C) \le \frac{1+\varepsilon'}{1-\varepsilon'}.$$

Proof. Assume first that $\|A_{\mathcal{H}} - C^{-1}\|_2 \le \varepsilon \|A_{\mathcal{H}}\|_2$. Since

$$\|I - (A_{\mathcal{H}} C)^{-1}\|_2 = \|(A_{\mathcal{H}} - C^{-1}) A_{\mathcal{H}}^{-1}\|_2 \le \varepsilon \, \|A_{\mathcal{H}}\|_2 \|A_{\mathcal{H}}^{-1}\|_2 = \varepsilon \operatorname{cond}_2(A_{\mathcal{H}}),$$

one can apply the estimates of the proof of Lemma 2.41 with $A_{\mathcal{H}} C$ replaced by $(A_{\mathcal{H}} C)^{-1}$.

If $\|A_{\mathcal{H}}^{-1} - C\|_2 \le \varepsilon \|A_{\mathcal{H}}^{-1}\|_2$, then

$$\|I - A_{\mathcal{H}} C\|_2 = \|A_{\mathcal{H}} (A_{\mathcal{H}}^{-1} - C)\|_2 \le \varepsilon \, \|A_{\mathcal{H}}\|_2 \|A_{\mathcal{H}}^{-1}\|_2 = \varepsilon \operatorname{cond}_2(A_{\mathcal{H}})$$

gives the assertion. □

The stronger condition $\varepsilon \operatorname{cond}_2(A_{\mathcal{H}}) < 1$ implies that ε has to go to zero if $A_{\mathcal{H}}$ is not well-conditioned. This, however, will not destroy the almost linear complexity since it will be seen that the complexity of the $\mathcal{H}$-matrix approximation depends logarithmically on the accuracy ε.

2.12.1.1 Clusters of Eigenvalues

From Theorem 2.40 it can also be seen that few small or large eigenvalues λ_i' and λ_i'' do not affect the rate of convergence. Therefore, the distribution of eigenvalues within the spectrum is even more important for the rate of convergence than the condition number, which depends only on the minimum and maximum eigenvalue of $A_{\mathcal{H}} C$. A faster convergence of PCG can be obtained by a condition on the distribution neglecting the size of the interval.

Definition 2.44. By $\gamma_A(\varepsilon)$ we denote the number of eigenvalues of $A \in \mathbb{R}^{n \times n}$ lying outside a disc of radius $\varepsilon > 0$ centered at the origin. The eigenvalues of a sequence of matrices $\{A_n\}_{n \in \mathbb{N}}$ are said to have a **cluster** at 0 if $\gamma_{A_n}(\varepsilon)$ is bounded independently of n. The eigenvalues of $\{A_n\}_{n \in \mathbb{N}}$ are said to have a cluster at $z \in \mathbb{C}$ if $\{A_n - z I_n\}_{n \in \mathbb{N}}$ has a cluster at 0.

If $A_{\mathcal{H}}$ and C are Hermitian positive definite and $A_{\mathcal{H}} C$ has a cluster at 1, then PCG converges super-linearly; i.e., for all sufficiently large n the residual in the kth step is bounded by $c\, q^k$ for all $0 < q < 1$. Why this super-linear convergence happens is explained in [9].

The following theorem states that for the existence of eigenvalue clusters the approximation C of $A_{\mathscr{H}}^{-1}$ does not have to be too accurate.

Theorem 2.45. *Let $\{A_n\}_{n\in\mathbb{N}} \subset \mathbb{R}^{n\times n}$ be a bounded sequence, i.e., $\|A_n\|_F \le c$, where c does not depend on n. Then both the singular values and the eigenvalues of $\{A_n\}_{n\in\mathbb{N}}$ cluster at 0.*

Proof. By $\nu_A(\varepsilon)$ we denote the number of singular values $\sigma_i(A)$, $i = 1,\dots,n$, of A lying outside a disc of radius $\varepsilon > 0$ centered at the origin. Assume that $\nu_A(\varepsilon) > c^2/\varepsilon^2$. Then

$$c^2 < \nu_A(\varepsilon)\varepsilon^2 \le \sum_{i=1}^{n} \sigma_i^2(A) = \|A\|_F^2,$$

which gives the contradiction. Hence, $\nu_A(\varepsilon) \le c^2/\varepsilon^2$.

In order to prove the same property for the eigenvalues, let $A = QTQ^H$ be Schur's form with unitary Q and upper triangular matrix T with the eigenvalues of A on its diagonal. The assertion follows from

$$\|A\|_F^2 = \|T\|_F^2 \ge \sum_{i=1}^{n} |\lambda_i(A)|^2$$

and the same arguments as above. □

Therefore, if for C it holds that

$$\|I - A_{\mathscr{H}}C\|_F \le c$$

with a constant $c > 0$ which does not depend on n, then the eigenvalues of $A_{\mathscr{H}}C$ cluster at 1 and PCG converges super-linearly. Note that in order to guarantee that C is positive definite, we can employ the stabilized rounded addition from Sect. 2.5.1 during the computations without any further assumption on c.

2.12.2 Non-Hermitian Coefficient Matrices

If $\mathscr{A}$ is not self-adjoint, then $A_{\mathscr{H}}$ cannot be expected to be Hermitian. In this case, not the spectral condition number of the coefficient matrix but the distance of a cluster of eigenvalues to the origin usually determines the convergence rate of appropriate Krylov subspace methods such as GMRes, BiCGStab, and MinRes. Nevertheless, a low-accuracy approximate inverse will be sufficient to guarantee a bounded number of iterations.

For the convergence of GMRes, for instance, the **numerical range**

$$\mathscr{F}(A_{\mathscr{H}}C) := \left\{x^H A_{\mathscr{H}}Cx : x \in \mathbb{C}^n,\ \|x\|_2 = 1\right\}$$

of $A_{\mathscr{H}}C$ is of particular importance. It is known (see [117]) that for the kth iterate x_k of GMRes applied to $Ax = b$ it holds that

$$\|b - A_{\mathscr{H}} x_k\|_2 \leq 2 \left(\frac{r}{|z|} \right)^k \|b\|_2$$

provided that $\mathscr{F}(A_{\mathscr{H}} C) \subset B_r(z)$, where $B_r(z)$ denotes the closed disc around z with radius r. A similar behavior can be observed for other non-Hermitian Krylov subspace methods. Condition (2.23) implies that $\mathscr{F}(A_{\mathscr{H}} C) \subset B_\varepsilon(1)$, which follows from

$$|x^H A_{\mathscr{H}} C x - 1| = |x^H (A_{\mathscr{H}} C - I) x| \leq \|I - A_{\mathscr{H}} C\|_2 \leq \varepsilon \quad \text{for all } x \in \mathbb{C}^n, \|x\|_2 = 1.$$

Therefore, (2.23) also leads to a problem-independent convergence

$$\|b - A_{\mathscr{H}} x_k\|_2 \leq 2\varepsilon^k \|b\|_2 \tag{2.27}$$

of GMRes.

In the following chapters these results will be used when solving problems arising from the discretization of integral operators and boundary value problems.

Chapter 3
Approximation of Discrete Integral Operators

In the following chapters we turn to the major concern of this book, the efficient numerical solution of elliptic boundary value problems

$$\mathscr{L}u = f \quad \text{in } \Omega \text{ (or } \Omega^c\text{)}, \tag{3.1a}$$
$$\gamma_0 u = g_D \quad \text{on } \Gamma_D, \tag{3.1b}$$
$$\gamma_1 u = g_N \quad \text{on } \Gamma_N \tag{3.1c}$$

on bounded Lipschitz domains $\Omega \subset \mathbb{R}^d$ or its complement $\Omega^c := \mathbb{R}^d \setminus \overline{\Omega}$. The operator $\mathscr{L}$ is a second order partial differential operator which may be scalar or a system of m operators

$$(\mathscr{L}u)_k = -\sum_{i,j=1}^{d}\sum_{\ell=1}^{m} \partial_i(c_{ij}^{k\ell}\partial_j + \gamma_i^{k\ell})u_\ell + \beta_j^{k\ell}\partial_j u_\ell + \delta^{k\ell}u_\ell, \quad k = 1,\dots,m, \tag{3.2}$$

with coefficient functions $c_{ij}^{k\ell}$, $\beta_j^{k\ell}$, $\gamma_i^{k\ell}$, and $\delta^{k\ell}$, $i,j = 1,\dots,d$, $k,\ell = 1,\dots,m$. Besides f, the Dirichlet data g_D and the Neumann data g_N are given on parts Γ_D and Γ_N of the boundary $\partial\Omega =: \Gamma = \overline{\Gamma}_D \cup \overline{\Gamma}_N$. In (3.1), γ_0 and γ_1 denote the Dirichlet trace and the conormal derivative

$$(\gamma_1 u)_k := \sum_{i,j=1}^{d}\sum_{\ell=1}^{m} n_i c_{ij}^{k\ell}\partial_j u_\ell \quad \text{on } \Gamma_N,$$

where $n(x)$ denotes the outer normal in the point $x \in \Gamma_N$. In addition to Dirichlet and Neumann conditions, also conditions of Robin type $\gamma_1 u - c\gamma_0 u = g_R$ with given g_R on $\Gamma_R \subset \Gamma$ may appear.

The differential operator $\mathscr{L}$ is assumed to be uniformly elliptic in the sense that the following **Legendre-Hadamard condition** (see [101]) holds

$$\sum_{i,j=1}^{d}\sum_{k,\ell=1}^{m} c_{ij}^{k\ell} v_i w_k v_j w_\ell \geq \lambda_{\mathscr{L}} \|v\|^2 \|w\|^2 \quad \text{for all } v \in \mathbb{R}^d,\ w \in \mathbb{R}^m, \tag{3.3}$$

M. Bebendorf, *Hierarchical Matrices*. Lecture Notes in Computational Science and Engineering 63,

where $\|\cdot\|$ denotes the Euclidean norm and $\lambda_{\mathscr{L}} > 0$ is a positive constant. Additionally, we assume that

$$\max_{x\in\Omega} |c_{ij}^{k\ell}(x)| \le \Lambda_{\mathscr{L}}, \quad i,j = 1,\dots,d, \quad k,\ell = 1,\dots,m.$$

Boundary value problems of type (3.1) are used to model problems arising in many fields of science such as stationary heat transfer, the computation of electromagnetic fields, and ideal flows. The easiest example of operators $\mathscr{L}$ is the **Laplacian** $\mathscr{L} = -\Delta$. An example for systems of boundary value problems are the **Lamé equations**

$$-\mu\vec{\Delta}u - (\lambda+\mu)\nabla\operatorname{div} u = f, \tag{3.4}$$

which arise from linear isotropic elasticity. Here, $\vec{\Delta}$ is a diagonal $d\times d$ matrix having the entries Δ, and $\lambda,\mu > 0$ denote the Lamé constants. Operator (3.4) is in the class of operators (3.2) with coefficients $c_{ij}^{k\ell} = \mu\delta_{ij}\delta_{k\ell} + (\lambda+\mu)\delta_{ik}\delta_{j\ell}$, where $\delta_{k\ell}$ denotes the Kronecker symbol. Therefore,

$$\sum_{i,j=1}^{d}\sum_{k,\ell=1}^{m} c_{ij}^{k\ell} v_i w_k v_j w_\ell = \mu\|v\|^2\|w\|^2 + (\lambda+\mu)(v\cdot w)^2 \ge \mu\|v\|^2\|w\|^2$$

such that the Lamé operator satisfies condition (3.3). Another example of systems of partial differential operators is the operator

$$\mathscr{L} := \operatorname{curl}\operatorname{curl} - \alpha\nabla\operatorname{div}, \quad \alpha > 0, \tag{3.5}$$

which arises from Maxwell's equations. Since $\operatorname{curl}\operatorname{curl} = -\vec{\Delta} + \nabla\operatorname{div}$, we obtain that

$$\mathscr{L} = -\vec{\Delta} + (1-\alpha)\nabla\operatorname{div}.$$

Hence, the **curl-curl operator** has a similar structure as the Lamé operator. As a consequence, $c_{ij}^{k\ell} = \delta_{ij}\delta_{k\ell} + (\alpha-1)\delta_{ik}\delta_{j\ell}$ gives

$$\sum_{i,j=1}^{d}\sum_{k,\ell=1}^{m} c_{ij}^{k\ell} v_i w_k v_j w_\ell = \|v\|^2\|w\|^2 + (\alpha-1)(v\cdot w)^2.$$

If $\alpha \ge 1$, then the previous expression is obviously bounded by $\|v\|^2\|w\|^2$ from below. In the other case, $0 < \alpha < 1$, we observe that

$$\begin{aligned}\|v\|^2\|w\|^2 + (\alpha-1)(v\cdot w)^2 &= \alpha\|v\|^2\|w\|^2 + (1-\alpha)[\|v\|^2\|w\|^2 - (v\cdot w)^2] \\ &\ge \alpha\|v\|^2\|w\|^2\end{aligned}$$

due to the Cauchy-Schwarz inequality.

The components of the solution of (3.1) are searched in **Sobolev spaces**

$$W^{k,p}(\Omega) := \{u \in L^p(\Omega) : \partial^\alpha u \in L^p(\Omega) \text{ for all } |\alpha| \le k\}, \quad k \in \mathbb{N}_0,\ p \in \mathbb{N},$$

where as usual by ∂^α we denote the weak derivative

$$\partial^\alpha = \left(\frac{\partial}{\partial x_1}\right)^{\alpha_1} \left(\frac{\partial}{\partial x_2}\right)^{\alpha_2} \cdots \left(\frac{\partial}{\partial x_d}\right)^{\alpha_d}$$

with the multi-index $\alpha \in \mathbb{N}_0$ and $|\alpha| = \alpha_1 + \cdots + \alpha_d$. We will also make use of the notations $x^\alpha = x_1^{\alpha_1} \cdot \ldots \cdot x_d^{\alpha_d}$ and $\alpha! = \alpha_1! \cdot \ldots \cdot \alpha_d!$. The set $W_0^{k,p}(\Omega)$ is defined as the closure of the set

$$C_0^\infty(\Omega) := \{u \in C^\infty(\Omega) : \operatorname{supp} u \subset \Omega\}$$

in $W^{k,p}(\Omega)$ with respect to the norm

$$\|u\|_{W^{k,p}(\Omega)} := \left(\sum_{|\alpha|\le k} \|\partial^\alpha u\|_{L^p(\Omega)}^p\right)^{1/p}.$$

Here, $\operatorname{supp} u := \overline{\{x \in \Omega : u(x) \neq 0\}}$ denotes the support of u in Ω. For $p = 2$ we obtain the Hilbert spaces $H^k(\Omega) := W^{k,2}(\Omega)$ and $H_0^k(\Omega) := W_0^{k,2}(\Omega)$ with the scalar product

$$(u,v)_{H^k(\Omega)} := \sum_{|\alpha|\le k} \int_\Omega \partial^\alpha u \, \partial^\alpha v \, \mathrm{d}x.$$

The Sobolev space $H^{-k}(\Omega)$ of negative order is defined as the dual space of $H_0^k(\Omega)$.

Sobolev spaces $H^s(\Gamma)$, $s \in \mathbb{R}$, on the boundary can be defined using parametrizations of Γ. The definition of $H^s(\Gamma)$, $|s| \le k$, therefore requires that the boundary Γ is in $C^{k-1,1}$. For a part $\Gamma_0 \subset \Gamma$ of the boundary Γ one can introduce Sobolev spaces $H^s(\Gamma_0)$ and $\tilde{H}^s(\Gamma_0)$, $s \ge 0$, by

$$H^s(\Gamma_0) := \{u|_{\Gamma_0} : u \in H^s(\Gamma)\} \quad \text{and} \quad \tilde{H}^s(\Gamma_0) := \{u|_{\Gamma_0} : u \in H^s(\Gamma), \operatorname{supp} u \subset \overline{\Gamma}_0\}$$

with the norm

$$\|u\|_{H^s(\Gamma_0)} := \inf\{\|\tilde{u}\|_{H^s(\Gamma)} : \tilde{u} \in H^s(\Gamma), \tilde{u}|_{\Gamma_0} = u\}.$$

Negative Sobolev spaces on the part Γ_0 of the boundary are defined again by duality

$$H^{-s}(\Gamma_0) := [\tilde{H}^s(\Gamma_0)]' \quad \text{and} \quad \tilde{H}^{-s}(\Gamma_0) := [H^s(\Gamma_0)]'.$$

For further properties of Sobolev spaces the reader is referred to [2].

Since (3.1) usually cannot be solved analytically, it is discretized by finite difference, finite element or finite volume methods; see [126]. The latter methods are particularly useful if the computational domain Ω is bounded and the operator $\mathscr{L}$ has nonsmooth coefficients. If exterior boundary value problems are to be solved, then the reformulation of the boundary value problem (3.1) as an integral equation on the boundary of Ω has several advantages. Boundary integral formulations do not require the discretization of unbounded domains as volume methods do. Although

a careful choice of artificial, absorbing boundary conditions can reduce the size of the computational domain, the finite element method may suffer from grid dispersion errors leading to spurious solutions. In addition, the generation of meshes for boundaries is much easier than meshing a volume. Due to the reduced spatial dimension, the number of degrees of freedom of boundary element discretizations is significantly smaller than the number of degrees of freedom in volume discretizations. A reformulation of (3.1) as a boundary integral equation, however, is possible only if $f = 0$ and if the coefficients of the operator are constant or at least sufficiently smooth. Piecewise constant coefficients can be handled by substructuring methods (see [243] for an overview) among which are the recently introduced BETI methods [174]. The advantages of both finite element (FE) and boundary element (BE) methods can be exploited by coupling them on unbounded domains; see [70].

While in Chap. 4 hierarchical matrices are applied to finite element discretizations, in this chapter we concentrate on the fast numerical treatment of boundary integral or more generally *non-local* operators. In Sect. 3.1 a brief overview of boundary element methods will be given in order to gain insight into the properties that can be exploited by efficient schemes. Due to the non-locality, discretizations of such operators are usually fully populated matrices such that at least n^2 operations are necessary to compute the matrix entries, which require n^2 units of storage. Here and in the rest of this book, n denotes the number of degrees of freedom. The latter property was a long-thought disadvantage of integral methods compared with the sparse structure of finite element matrices, for instance.

In order to avoid dense matrices, different approaches have been introduced. For rotational invariant geometries, the arising matrices are likely to have Toeplitz structure, which can be exploited (cf. [210]) by algorithms based on the fast Fourier transform. However, such structures are not present in general; for the approximation by a sum of circulant and low-rank matrices see [33]. The pre-corrected FFT method [204] appropriately modifies the discrete operator such that the grid is replaced by a regular one, on which translational invariant kernels lead to Toeplitz matrices.

Fast methods which can be applied in a quite general setting usually rely on approximation. Since the discrete solution cannot be expected to be more accurate than the discretization error, we can admit additional errors which are of the same order. Existing approximate methods can be subdivided into two classes. If the underlying geometry can be described by a small number of smooth maps, wavelet techniques can be used for compressing the discrete operators; see [40, 3, 73, 255, 232, 256, 171]. Instead of the usual ansatz and trial functions these methods employ wavelet basis functions for the discretization of the integral operator. Although the latter approach leads to sparse and asymptotically well-conditioned coefficient matrices, the efficient computation of the remaining entries is still subject of current research. Improvements of the applicability of wavelets to unstructured grids and complicated geometries have been reported in [251, 142].

The second class of approximate methods exploits the local existence of degenerate approximants to the kernel function κ of the integral operator; i.e.,

$$\kappa(x,y) \approx \sum_{\ell=1}^{k} u_\ell(x) v_\ell(y), \tag{3.6}$$

where k is a small number. This idea originating from the Barnes-Hut algorithm [14] was refined in the fast multipole method (FMM) [215, 121, 118, 122] and the panel clustering algorithm [138, 226, 139]. For an overview on many papers on the fast multipole method see [192]. A generalization of the fast multipole method to *asymptotically smooth* kernel functions is proposed in [51, 50, 103]. Instead of spherical harmonics, the more flexible algebraic polynomials are used. In Sect. 3.2 it will be proved that the kernel functions arising from boundary integral formulations of (3.1) is asymptotically smooth provided that the Legendre-Hadamard condition (3.3) is satisfied. This will be done by proving a Caccioppoli-type estimate for the arising kernel functions. A similar result will be proved also for the biharmonic operator, which is a forth order elliptic partial differential operator.

Section 3.3 is devoted to the construction of degenerate kernel approximants. Asymptotically smooth kernel functions are shown to admit approximations of type (3.6) on pairs of domains satisfying the admissibility condition (1.9) from Example 1.13. We will present the standard construction, which is based on polynomial interpolation, and a second technique which is not based on any specific system of interpolation but uses restrictions of the kernel function for its approximation. Note that kernel approximation methods can also be applied to integral operators which do not stem from the reformulation of boundary value problems. The kernel function of many integral operators such as the Gauß transform possess properties which makes them well-suited for these methods. Further examples will be given in Sect. 3.3.

The mentioned methods provide an efficient (approximate) representation of the discrete operator for multiplying it by a vector with logarithmic-linear complexity. Hence, they are well-suited for iterative solvers such as Krylov subspace methods. In recent years, (3.6) has been regarded more from an algebraic point of view. On the discrete level, property (3.6) means that admissible blocks of the discrete operator can be approximated by matrices of low rank; cf. Sect. 3.3. This gave rise to the mosaic-skeleton method [252, 111, 110] and to hierarchical matrices [127, 133]. We have already pointed out in Sect. 2.11 that $\mathscr{H}^2$-matrices [136] are an algebraic generalization of the fast multipole method which in particular can handle polynomial based fast multipole methods; see [130]. Using $\mathscr{H}^2$-matrices with variable order techniques (cf. [227, 250, 184]), it is possible to treat discrete integral operators with exactly linear complexity. Variable order approximations can, however, be applied only in very special situations. Existence results and numerical experiments (see [227, 136, 184, 250]) concentrate on pseudo-differential operators of order zero. Throughout this book we will therefore use $\mathscr{H}$-matrices for the efficient treatment of the arising matrices.

Based on the results of Sect. 3.3, in Sect. 3.4 we present an efficient and easy to use method, the *adaptive cross approximation* (ACA) algorithm [17, 32, 26], which aims at the construction of low-rank approximants from few of the original entries of a discrete integral operator instead of approximating its kernel function. Since the

convergence will be proved for asymptotically smooth functions, this method can be applied in particular to the integral formulation of any elliptic boundary value problem. The parallelization of this technique will be considered too. It will be proved that ACA provides quasi-optimal results in the sense that, up to constants, the approximation error is smaller than in any other system of approximation including polynomials and spherical harmonics which multipole methods are based on. We report the results of applying this technique to the Laplacian on complicated domains and to problems from elasticity. In Sect. 3.5 we present a method which is based on recompression of the approximation obtained by ACA. Note this recompression technique differs significantly from the algebraic technique presented in Sect. 2.6. The latter method is able to improve constants in the asymptotic complexity, while the former allows to obtain the asymptotic complexity of $\mathscr{H}^2$-matrix and preserves the convenient properties of ACA.

The application of ACA to realistic and challenging problems will be considered in Sect. 3.6. Once an efficient representation of a discrete integral operator has been constructed, it often appears as the coefficient matrix of a linear system. The solution is usually done by an iterative Krylov method, the efficiency of which is determined by spectral properties such as the distribution of the eigenvalues in the normal case and the numerical range of the coefficient matrix. Since the spectrum of most of the arising discrete operators is unbounded, preconditioning is necessary. For this purpose, approximate *LU* decompositions (see Sect. 2.9) can be computed, which can serve as preconditioners. A major practical advantage of such preconditioners is that they can be computed directly from the coefficient matrix by a black-box procedure.

Instead of constructing efficient methods which are based on applying the discrete operator to a vector in an efficient way, in [66, 151, 178] fast direct methods have been proposed in two spatial dimensions. Although the hierarchical *LU* decomposition could be employed as a direct solver also in general dimensions, we will use it for preconditioning purposes only. The usage as a direct solver, however, should be preferred in situations involving multiple right-hand sides.

3.1 Boundary Integral Formulations

Assume that $g_D \in H^{1/2}(\Gamma_D)$ and $g_N \in H^{-1/2}(\Gamma_N)$. Then the solution u of (3.1) can be expected to be uniquely determined by the conditions on the boundary Γ provided that the Dirichlet part has positive measure. Pure Neumann problems can also be treated if additional constraints are imposed. Under these conditions the boundary value problem (3.1) with $f = 0$ can be reformulated as integral equations on the boundary Γ; cf. [229, 65, 182, 242, 228]. For simplicity, we restrict ourselves to the leading part of the operator $\mathscr{L}$; i.e., we consider operators

$$(\mathscr{L}u)_k = -\sum_{i,j=1}^{d}\sum_{\ell=1}^{m}\partial_i(c_{ij}^{k\ell}\partial_j)u_\ell, \quad k = 1,\dots,m.$$

Assume that for $\mathscr{L}$ a **singularity function** S, i.e., an $m \times m$ matrix of functions satisfying

$$\mathscr{L}S = \delta I_m \quad \text{in } \mathbb{R}^d,$$

can be found. Here, δ denotes Dirac's δ. The existence of S can be guaranteed at least for operators $\mathscr{L}$ with constant coefficients; cf. [164]. Since S will appear as part of the kernel function of the resulting integral operators, it is advantageous to know S explicitly. Singularity functions for many operators can be found in [170].

The boundary element method is applied to various partial differential operators such as the Lamé equations and the **Helmholtz operator** $-\Delta - \omega^2$, $\omega \in \mathbb{R}$, arising from acoustics. The reformulation of the different types of boundary value problems as boundary integral equations is based on common principles; cf. [71]. For simplicity we explain briefly how the boundary integral equations are derived if $\mathscr{L} = -\Delta$ is the Laplacian. In this case, the conormal derivative γ_1 coincides with the normal derivative ∂_ν. The respective integral equations for the Lamé operator can be found in (3.74).

Making use of **Green's representation formula** (cf. [182])

$$\pm u(x) = (\mathscr{V}t)(x) - (\mathscr{K}u)(x), \quad x \in \Omega \ (x \in \Omega^c), \tag{3.7}$$

the solution u of (3.1) can be represented in Ω or its exterior Ω^c by the Dirichlet data $u(x)$ and the Neumann data $t(x) := \partial_\nu u(x)$ for $x \in \Gamma$. In (3.7), $\mathscr{V}$ denotes the **single-layer potential operator**

$$(\mathscr{V}w)(y) := \int_\Gamma S(x-y)w(x)\,\mathrm{d}s_x, \quad y \in \mathbb{R}^d,$$

acting on w on the boundary $\Gamma = \Gamma_D \cup \Gamma_N$. $\mathscr{K}$ is the **double-layer potential operator**

$$(\mathscr{K}w)(y) := \int_\Gamma w(x)\partial_{\nu_x} S(x-y)\,\mathrm{d}s_x, \quad y \in \mathbb{R}^d.$$

Here,

$$S(x) = \begin{cases} -\frac{1}{2}|x|, & d = 1, \\ -\frac{1}{2\pi}\ln\|x\|, & d = 2, \\ \frac{1}{(d-1)\omega_d'}\|x\|^{2-d}, & d \geq 3 \end{cases} \tag{3.8}$$

denotes the singularity function of the Laplacian and ω_d' is the surface measure of the unit sphere in $\mathbb{R}^d$.

Applying the trace operators γ_0 and γ_1 to the representation formula (3.7) together with the **jump relations**

$$\begin{aligned} \partial_\nu(\mathscr{V}w)(y) \quad &\to \quad (\mathscr{K}'w)(y_0) \pm \frac{1}{2}w(y_0), \\ (\mathscr{K}w)(y) \quad &\to \quad (\mathscr{K}w)(y_0) \mp \frac{1}{2}w(y_0) \end{aligned}$$

for $y \to y_0 \in \Gamma$, $y \in \Omega$ ($y \in \Omega^c$), yields the following system of boundary integral equations on Γ

$$\begin{bmatrix} \gamma_0 u \\ \gamma_1 u \end{bmatrix} = \begin{bmatrix} \pm\frac{1}{2}\mathscr{I} - \mathscr{K} & \mathscr{V} \\ \mathscr{D} & \pm\frac{1}{2}\mathscr{I} + \mathscr{K}' \end{bmatrix} \begin{bmatrix} \gamma_0 u \\ \gamma_1 u \end{bmatrix} \tag{3.9}$$

in which

$$(\mathscr{K}' w)(y) := \int_\Gamma w(x)\, \partial_{\nu_y} S(x-y)\, \mathrm{d}s_x, \quad y \in \Gamma,$$

denotes the adjoint of $\mathscr{K}$ and $\mathscr{D}$ is the **hypersingular operator** which results from applying the negative Neumann trace γ_1 to the double-layer potential operator. It is known (see, for instance, [182]) that $\mathscr{V} : \tilde{H}^{-1/2}(\Gamma_D) \to H^{1/2}(\Gamma_D)$ is continuous and $\tilde{H}^{-1/2}(\Gamma_D)$-**coercive**; i.e.,

$$\langle \mathscr{V} w, w \rangle_{L^2(\Gamma_D)} \geq c_{\mathscr{V}} \|w\|^2_{\tilde{H}^{-1/2}(\Gamma_D)} \quad \text{for all } w \in \tilde{H}^{-1/2}(\Gamma_D). \tag{3.10}$$

Furthermore, $\mathscr{D} : \tilde{H}^{1/2}(\Gamma_N) \to H^{-1/2}(\Gamma_N)$ is continuous and $\tilde{H}^{1/2}(\Gamma_N)$-coercive; i.e.,

$$\langle \mathscr{D} w, w \rangle_{L^2(\Gamma_N)} \geq c_{\mathscr{D}} \|w\|^2_{\tilde{H}^{1/2}(\Gamma_N)} \quad \text{for all } w \in \tilde{H}^{1/2}(\Gamma_N) \tag{3.11}$$

provided that Γ_D has positive measure. The operator $\mathscr{K} : H^{1/2}(\Gamma) \to H^{1/2}(\Gamma)$ is continuous; i.e.,

$$\langle \mathscr{K} v, w \rangle_{L^2(\Gamma)} \leq c_{\mathscr{K}} \|v\|_{H^{1/2}(\Gamma)} \|w\|_{H^{1/2}(\Gamma)} \quad \text{for all } v, w \in H^{1/2}(\Gamma). \tag{3.12}$$

Since u is known only on Γ_D and t is known only on Γ_N, we have to complete the Cauchy data $[\gamma_0 u, \gamma_1 u]$. Let $\tilde{g}_D$ and $\tilde{g}_N$ denote the canonical extensions of g_D and g_N to Γ. Setting $\tilde{u} := u - \tilde{g}_D$ and $\tilde{t} := t - \tilde{g}_N$, we have to compute $\tilde{u} \in \tilde{H}^{1/2}(\Gamma_N)$ and $\tilde{t} \in \tilde{H}^{-1/2}(\Gamma_D)$. Since $\tilde{u} = 0$ on Γ_D and $\tilde{t} = 0$ on Γ_N, from (3.9) we obtain

$$-\mathscr{V}\tilde{t} + \mathscr{K}\tilde{u} = \mathscr{V}\tilde{g}_N - \left(\frac{1}{2}\mathscr{I} + \mathscr{K}\right)\tilde{g}_D \quad \text{on } \Gamma_D, \tag{3.13a}$$

$$\mathscr{K}'\tilde{t} + \mathscr{D}\tilde{u} = \left(\frac{1}{2}\mathscr{I} - \mathscr{K}'\right)\tilde{g}_N - \mathscr{D}\tilde{g}_D \quad \text{on } \Gamma_N \tag{3.13b}$$

for the inner boundary value problem. The previous system of integral equations is referred to as the **symmetric boundary integral formulation** of the mixed boundary value problem (3.1); see [236, 70, 237]. It is uniquely solvable for non-vanishing Dirichlet boundaries Γ_D; cf. [182].

Remark 3.1. For the approximation by degenerate functions it will be important that the kernel functions κ of the arising integral operators $\mathscr{A} : V \to V'$ of the form

$$(\mathscr{A} v)(y) := \int_\Gamma \kappa(x,y) v(x)\, \mathrm{d}s_x \tag{3.14}$$

consist of normal derivatives of the singularity function S.

After approximating the manifold Γ by possibly curved triangles and quadrilaterals ($d = 3$) or segments ($d = 2$), each operator $\lambda\mathscr{I} + \mathscr{A}$ appearing in (3.13) is discretized as $\lambda M + A$, where the mass matrix M and the stiffness matrix A depend on the respective discretization method being used. Let φ_j, $j \in J$, be a basis of the finite-dimensional ansatz space $V_h \subset V$; i.e., the solution u_h is sought of the form $u_h = \sum_{j\in J} u_j \varphi_j$. In addition to piecewise linears also piecewise constant basis functions are commonly used together with one of the following three discretization methods.

(i) **Galerkin method**: Let $W_h \subset V'$ be a finite-dimensional trial space with associated basis ψ_i, $i \in I$. $(\lambda\mathscr{I} + \mathscr{A})u_h = g$ is tested in variational form

$$\int_\Gamma \lambda u_h \psi_i + (\mathscr{A}u_h)\psi_i \,\mathrm{d}s = \int_\Gamma g\psi_i \,\mathrm{d}s, \quad i \in I.$$

In this case, $m_{ij} = (\varphi_j, \psi_i)_{L^2(\Gamma)}$ and

$$a_{ij} = \int_\Gamma \int_\Gamma \kappa(x,y)\psi_i(y)\varphi_j(x) \,\mathrm{d}s_x \,\mathrm{d}s_y, \quad i \in I,\ j \in J.$$

(ii) **Collocation method**: The boundary integral equation is tested at collocation points $y_i \in \Gamma$, $i \in I$. Then $m_{ij} = \varphi_j(y_i)$ and

$$a_{ij} = \int_\Gamma \kappa(x,y_i)\varphi_j(x) \,\mathrm{d}s_x, \quad i \in I,\ j \in J.$$

(iii) **Nyström method**: In addition to testing the equation at collocation points $y_i \in \Gamma$, $i \in I$, the integral in (3.14) is replaced by a quadrature rule $Q(f) = \sum_{i\in I} \omega_i f(y_i')$ with weights $\omega_i \in \mathbb{R}$ and nodes $y_i' \in \Gamma$ such that $\varphi_j(y_i') = \delta_{ij}$. Then $m_{ij} = \varphi_j(y_i)$ and

$$a_{ij} = \omega_j \kappa(y_j', y_i), \quad i \in I,\ j \in J.$$

Note that the evaluation of the Coulomb potential

$$\sum_{j\in J} \frac{q_j}{|x - y_j|}$$

of point sources q_j located at points y_j, $j \in J$, can be regarded as the matrix-vector multiplication of a Nyström matrix with the vector $q := (q_j)_{j=1,\ldots,n}$.

In all three cases, M is sparse and A is a fully populated matrix. Since M does not produce any numerical difficulties, we will concentrate on A. The entries of M can be added to the non-admissible blocks of an approximant of A, because M vanishes on admissible blocks. For the discretization A of operators $\mathscr{A}$ it may happen that $|I| \neq |J|$. This is, for instance, the case if in the Galerkin method the ansatz space consists of piecewise linears and for the trial space piecewise constants are used.

Although the above discretization methods have principle differences, they can be treated in a common way when approximating the resulting matrices by data-sparse

methods. If $t \subset I$ and $s \subset J$, we define the following linear operators $\Lambda_{1,t} : L^2(\Gamma) \to \mathbb{R}^t$, $\Lambda_{2,s} : L^2(\Gamma) \to \mathbb{R}^s$. For $i \in t$ and $j \in s$ let

$$\begin{aligned}
&(\Lambda_{1,t} f)_i = \int_\Gamma f(x)\psi_i(x)\,\mathrm{d}s_x, && (\Lambda_{2,s} f)_j = \int_\Gamma f(x)\varphi_j(x)\,\mathrm{d}s_x && \text{Galerkin method,}\\
&(\Lambda_{1,t} f)_i = f(y_i), && (\Lambda_{2,s} f)_j = \int_\Gamma f(x)\varphi_j(x)\,\mathrm{d}s_x && \text{collocation method,}\\
&(\Lambda_{1,t} f)_i = f(y_i), && (\Lambda_{2,s} f)_j = \omega_j f(y'_j) && \text{Nyström method.}
\end{aligned}$$

With this notation, each block A_{ts} of the stiffness matrix $A \in \mathbb{R}^{I\times J}$ takes the form

$$A_{ts} = \Lambda_{1,t}\mathscr{A}\Lambda_{2,s}^*, \tag{3.15}$$

where $\Lambda_{2,s}^* : \mathbb{R}^s \to L^2(\Gamma)$ is the adjoint of $\Lambda_{2,s} : L^2(\Gamma) \to \mathbb{R}^s$ defined by

$$(\Lambda_{2,s}^* z, f)_{L^2(\Gamma)} = z^T(\Lambda_{2,s} f) \quad \text{for all } z \in \mathbb{R}^s,\ f \in L^2(\Gamma).$$

Additionally, we define the supports of $\Lambda : L^2(\Gamma) \to \mathbb{R}^t$ by

$$\operatorname{supp}\Lambda = \Gamma \setminus G,$$

where G is the largest open set such that $\Lambda\varphi = 0$ for all φ satisfying $\operatorname{supp}\varphi \subset G$. With this definition we set

$$Y_i := \operatorname{supp}\Lambda_{1,i},\ i \in I, \qquad X_j := \operatorname{supp}\Lambda_{2,j},\ j \in J.$$

The operators $\Lambda_{1,i}$ and $\Lambda_{2,j}$ project $\mathscr{A}$ onto one-dimensional subspaces. For the purpose of data-sparse approximation the following localization effect of $\Lambda_{1,i}$ and $\Lambda_{2,j}$ is even more important. The latter operators guarantee that for computing the entry a_{ij} the kernel function κ is evaluated on $X_j \times Y_i$. Hence, for the sub-block A_{ts} we have to evaluate κ on $X_s := \bigcup_{j\in s} X_j$ and $Y_t := \bigcup_{i\in t} Y_i$. For collocation methods, for instance, this means that κ is evaluated on the supports $X_j = \{y_j\}$ and $Y_i = \operatorname{supp}\psi_i$ of $\Lambda_{2,j}$ and $\Lambda_{1,i}$. The localizing effect is obvious for $\Lambda_{2,j}$. The latter property for $\Lambda_{1,i}$ results from the fact that FE basis functions ψ_i are locally supported. In the following we will therefore refer to $\Lambda_{1,i}$ and $\Lambda_{2,j}$ as **localizers**.

As an example for the application of discretization methods to (3.13) we consider the case that the Galerkin method is used. The partially known Neumann data is searched of the form $t_h := \sum_{i=1}^{n'} t_i\psi_i$, where ψ_i are piecewise constants such that $W_h := \operatorname{span}\{\psi_i, i = 1,\dots,n'\} \subset H^{-1/2}(\Gamma)$. Furthermore, let $V_h := \operatorname{span}\{\varphi_j, j = 1,\dots,n\}$ be made of piecewise linears. The discrete variational formulation of (3.13) leads to the following algebraic system of equations for the unknown coefficients $u \in \mathbb{R}^{n_D}$ and $t \in \mathbb{R}^{n'_N}$ of u_h and t_h

$$\begin{bmatrix} -V & K \\ K^T & D \end{bmatrix} \begin{bmatrix} t \\ u \end{bmatrix} = \begin{bmatrix} V & -\frac{1}{2}M - K \\ \frac{1}{2}M - K^T & -D \end{bmatrix} \begin{bmatrix} \tilde{g}_N \\ \tilde{g}_D \end{bmatrix} =: \begin{bmatrix} f_N \\ f_D \end{bmatrix}. \tag{3.16}$$

The entries of the above matrices are

$$V_{k\ell} = (\mathscr{V}\psi_\ell, \psi_k)_{L^2}, \quad K_{kj} = (\mathscr{K}\varphi_j, \psi_k)_{L^2}, \quad D_{ij} = (\mathscr{D}\varphi_j, \varphi_i)_{L^2},$$

where $k, \ell = 1, \ldots, n'_N$ and $i, j = 1, \ldots, n_D$.

The convergence analysis of the above discretization methods is a generalization of the finite element analysis; cf. [259, 260]. Let $(\tilde{u}, \tilde{t}) \in H^2(\Gamma_N) \times H^1(\Gamma_D)$, be the solution of (3.13). In the described setting the convergence of u_h and t_h against $\tilde{u}$ and $\tilde{t}$ can be completely analyzed for $h \to 0$. Using Céa's lemma it can be shown that

$$\|\tilde{u} - u_h\|^2_{H^{1/2}(\Gamma_N)} + \|\tilde{t} - t_h\|^2_{H^{-1/2}(\Gamma_D)} \leq ch^3 \left(\|\tilde{u}\|^2_{H^2(\Gamma_N)} + \|\tilde{t}\|^2_{H^1(\Gamma_D)} \right).$$

Hence, the discrete solution (u_h, t_h) convergences for decreasing mesh size $h \to 0$ against the continuous solution $(\tilde{u}, \tilde{t})$ of (3.13); cf. [182]. Green's representation formula (3.7) can be used to generate an approximation of u from the computed approximate Cauchy data (u_h, t_h).

3.2 Asymptotic Smoothness of Kernel Functions

In this section we will lay theoretical ground to the efficient treatment of systems of partial differential operators

$$(\mathscr{L}u)_k = -\sum_{i,j=1}^{d} \sum_{\ell=1}^{m} \partial_i (c_{ij}^{k\ell} \partial_j) u_\ell + \delta u_k, \quad k = 1, \ldots, m, \tag{3.17}$$

with constant coefficients $c_{ij}^{k\ell}$ and δ satisfying the Legendre-Hadamard condition (3.3). We will prove that the associated singularity functions are asymptotically smooth.

Definition 3.2. A function $\kappa : \Omega \times \mathbb{R}^d \to \mathbb{R}$ satisfying $\kappa(x, \cdot) \in C^\infty(\mathbb{R}^d \setminus \{x\})$ for all $x \in \Omega$ is called **asymptotically smooth** in Ω with respect to y if constants c and γ can be found such that for all $x \in \Omega$ and all $\alpha \in \mathbb{N}_0^d$

$$|\partial_y^\alpha \kappa(x, y)| \leq cp!\gamma^p \frac{|\kappa(x, y)|}{\|x - y\|^p} \quad \text{for all } y \in \mathbb{R}^d \setminus \{x\}, \tag{3.18}$$

where $p = |\alpha|$.

Instead of (3.18) sometimes (see, for instance, [49]) the condition

$$|\partial_y^\alpha \kappa(x, y)| \leq cp!\gamma^p \|x - y\|^{-s-p} \quad \text{for all } x \neq y \tag{3.19}$$

on the derivatives of κ is used with some $s \in \mathbb{R}$. In the wavelet community the latter condition is referred to as the **Calderón-Zygmund property**. Note that conditions (3.18) and (3.19) are equivalent if κ has an algebraic singularity for $x = y$. Condition (3.18), however, better accounts for logarithmic singularities as they often appear for

$d = 2$. For instance, the singularity function $S(x) = \|x\|^2 \log \|x\|$ of the **biharmonic operator** Δ^2 (cf. [58]) satisfies (3.18) as we shall see, but it satisfies the alternative condition (3.19) only from the third derivative on.

Let $\mathfrak{F} : L^2(\Omega) \to L^2(\Omega)$ denote the Fourier transform defined by

$$(\mathfrak{F}v)(x) = \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} e^{-\mathrm{i}x\cdot y} v(y)\,\mathrm{d}y,$$

which is normalized such that $\mathfrak{F}^* = \mathfrak{F}^{-1}$. From $\mathfrak{F}[\partial^\alpha u](\xi) = \xi^\alpha (2\pi\mathrm{i})^{|\alpha|} \mathfrak{F}u(\xi)$, it follows that

$$-\partial_i c_{ij}^{k\ell} \partial_j u_\ell = -\mathfrak{F}^* \mathfrak{F}(\partial_i c_{ij}^{k\ell} \partial_j u_\ell) = (2\pi)^2 \mathfrak{F}^* (c_{ij}^{k\ell} \xi_i \xi_j \mathfrak{F} u_\ell)$$

and using (3.3) for $u \in [H_0^1(\Omega)]^m$ together with $\|\mathfrak{F}u\|_{L^2(\Omega)} = \|u\|_{L^2(\Omega)}$ one has

$$\sum_{i,j=1}^{d} \sum_{k,\ell=1}^{m} \int_\Omega c_{ij}^{k\ell} \partial_i u_k \partial_j u_\ell \,\mathrm{d}x = (2\pi)^2 \sum_{i,j=1}^{d} \sum_{k,\ell=1}^{m} \int_\Omega c_{ij}^{k\ell} \xi_i \xi_j \mathfrak{F} u_k \mathfrak{F} u_\ell \,\mathrm{d}x \tag{3.20a}$$

$$\geq (2\pi)^2 \lambda_{\mathscr{L}} \sum_{i=1}^{d} \sum_{\ell=1}^{m} \int_\Omega |\xi_i \mathfrak{F} u_\ell|^2 \,\mathrm{d}x \tag{3.20b}$$

$$= \lambda_{\mathscr{L}} \sum_{i=1}^{d} \sum_{\ell=1}^{m} \int_\Omega |\mathfrak{F}(\partial_i u_\ell)|^2 \,\mathrm{d}x \tag{3.20c}$$

$$= \lambda_{\mathscr{L}} \sum_{i=1}^{d} \sum_{\ell=1}^{m} \|\partial_i u_\ell\|_{L^2}^2 = \lambda_{\mathscr{L}} \|\mathfrak{J}(u)\|_{L^2}^2, \tag{3.20d}$$

where $\mathfrak{J}(u) \in \mathbb{R}^{m\times d}$ denotes the Jacobian of u.

The previous estimate does not hold for variable coefficients $c_{ij}^{k\ell}$. In this case, either one has to consider operators $(\mathscr{L}u)_k = -\sum_{i,j=1}^{d} \sum_{\ell=1}^{m} c_{ij}^{k\ell} \partial_i \partial_j u + \delta u_k$ instead of the divergence form (3.17) or the stronger condition

$$\sum_{i,j=1}^{d} \sum_{k,\ell=1}^{m} c_{ij}^{k\ell} v_{ik} v_{j\ell} \geq \lambda_{\mathscr{L}} \|v\|^2 \quad \text{for all } v \in \mathbb{R}^{dm} \tag{3.21}$$

has to be imposed instead of the Legendre-Hadamard condition (3.3). It is easily verified that the curl-curl operator (3.5) satisfies (3.21) only for $\alpha \geq 1$.

In order to prove asymptotic smoothness, we first derive the following Caccioppoli-type inequality with the usual technique, which uses cut-off functions. This equation embodies the so-called **interior regularity** of elliptic problems. Let $D \subset \mathbb{R}^d$ be a domain having an intersection with Ω that has positive measure.

Lemma 3.3. *Assume that $u \in [H^1(D)]^m$ is a weak solution of $\mathscr{L}u = 0$ in $D \cap \Omega$ and $u = 0$ in $D \setminus \Omega$. Then for any compact set $K \subset D$ it holds that*

$$\|\mathfrak{J}(u)\|_{L^2(K)} \leq \frac{c_{\mathscr{L}}}{\sigma} \|u\|_{L^2(D)},$$

where

$$c_{\mathscr{L}} = \sqrt{8d^2m\frac{\Lambda_{\mathscr{L}}}{\lambda_{\mathscr{L}}}\left(1+2dm\frac{\Lambda_{\mathscr{L}}}{\lambda_{\mathscr{L}}}\right)+2\frac{\sigma^2}{\lambda_{\mathscr{L}}}|\delta|} \quad \textit{and} \quad \sigma = \operatorname{dist}(K,\partial D).$$

Proof. Let $\eta \in C^1(D)$ satisfy $0 \le \eta \le 1$, $\eta = 1$ in K, $\eta = 0$ in a neighborhood of ∂D and $|\partial_i \eta| \le 2/\sigma$, $i = 1,\dots,d$, in D. With $u \in [H^1(\Omega)]^m$ it follows that $\eta^2 u \in [H_0^1(D\cap\Omega)]^m$. From

$$\partial_i(\eta^2 u_k)\partial_j u_\ell = \partial_i(\eta u_k)\partial_j(\eta u_\ell) + u_k\partial_i\eta\,\partial_j(\eta u_\ell) - u_\ell\partial_i(\eta u_k)\partial_j\eta - u_k u_\ell\partial_i\eta\partial_j\eta$$

we obtain using $\int_D \eta^2 u_k(\mathscr{L}u)_k\,\mathrm{d}x = 0$ that for any $\varepsilon > 0$

$$\begin{aligned}
&\sum_{i,j=1}^{d}\sum_{k,\ell=1}^{m}\int_\Omega c_{ij}^{k\ell}\partial_i(\eta u_k)\partial_j(\eta u_\ell)\,\mathrm{d}x + \delta\eta^2 u_k u_\ell \\
&\quad\le \Lambda_{\mathscr{L}}\sum_{i,j=1}^{d}\sum_{k,\ell=1}^{m}\int_\Omega |u_k||u_\ell||\partial_i\eta||\partial_j\eta| + 2|u_k||\partial_i\eta||\partial_j(\eta u_\ell)|\,\mathrm{d}x \\
&\quad\le 4\Lambda_{\mathscr{L}}\frac{d}{\sigma}\sum_{j=1}^{d}\sum_{k,\ell=1}^{m}\int_\Omega \frac{1}{\sigma}|u_k||u_\ell| + |u_k||\partial_j(\eta u_\ell)|\,\mathrm{d}x \\
&\quad\le 4\Lambda_{\mathscr{L}}\frac{d}{\sigma}\left(m\frac{d}{\sigma}\|u\|_{L^2}^2 + \sum_{j=1}^{d}\sum_{k,\ell=1}^{m}\left(\int_\Omega |u_k|^2\,\mathrm{d}x\right)^{1/2}\left(\int_\Omega |\partial_j(\eta u_\ell)|^2\,\mathrm{d}x\right)^{1/2}\right) \\
&\quad\le 2\Lambda_{\mathscr{L}}\frac{dm}{\sigma}\left(2\frac{d}{\sigma}\|u\|_{L^2}^2 + \frac{d}{\varepsilon}\sum_{k=1}^{m}\int_\Omega |u_k|^2\,\mathrm{d}x + \varepsilon\sum_{j=1}^{d}\sum_{\ell=1}^{m}\int_\Omega |\partial_j(\eta u_\ell)|^2\,\mathrm{d}x\right) \\
&\quad= 2\Lambda_{\mathscr{L}}\frac{d^2m}{\sigma^2}\left(2+\frac{\sigma}{\varepsilon}\right)\|u\|_{L^2(\Omega)}^2 + 2\Lambda_{\mathscr{L}}\frac{dm}{\sigma}\varepsilon\int_\Omega \|\mathfrak{J}(\eta u)\|_F^2\,\mathrm{d}x,
\end{aligned}$$

where we have used that $2ab \le a^2/\varepsilon + \varepsilon b^2$ for all $a,b \in \mathbb{R}$. From (3.20) we obtain

$$\begin{aligned}
\lambda_{\mathscr{L}}\int_D \|\mathfrak{J}(\eta u)\|_F^2\,\mathrm{d}x \le &\left(2\frac{d^2m\Lambda_{\mathscr{L}}}{\sigma^2}\left(2+\frac{\sigma}{\varepsilon}\right)+|\delta|\right)\|u\|_{L^2(D)}^2 \\
&+ 2\frac{dm\Lambda_{\mathscr{L}}}{\sigma}\varepsilon\int_D \|\mathfrak{J}(\eta u)\|_F^2\,\mathrm{d}x.
\end{aligned}$$

This leads to

$$\|\mathfrak{J}(u)\|_{L^2(K)}^2 \le \|\mathfrak{J}(\eta u)\|_{L^2(D)}^2 \le \frac{1}{\sigma^2}\frac{2d^2m\Lambda_{\mathscr{L}}(2+\sigma/\varepsilon)+\sigma^2|\delta|}{\lambda_{\mathscr{L}} - 2dm\Lambda_{\mathscr{L}}\varepsilon/\sigma}\|u\|_{L^2(D)}^2.$$

Choosing $\varepsilon = \lambda_{\mathscr{L}}\sigma/(4dm\Lambda_{\mathscr{L}})$ gives the desired result. □

Remark 3.4. From the previous proof it can be seen that non-negative δ do not enter the constant $c_{\mathscr{L}}$.

In order to derive pointwise estimates, we need the following estimate for u satisfying $\mathscr{L}u = 0$ in $B_r(x) \subset \mathbb{R}^d$ and $k \in \mathbb{N}$

$$\|u\|_{H^k(B_\rho(x))} \le c(k,\rho,r)\,\|u\|_{L^2(B_r(x))} \quad \text{for all } 0 < \rho < r, \tag{3.22}$$

where c depends on the coefficients of $\mathscr{L}$. Estimate (3.22) can be derived by applying Lemma 3.3 iteratively on a sequence of balls $B_{r_\ell}(x)$, $\ell = 1,\dots,k$. Due to the Sobolev embedding theorem, (3.22) states that $\mathscr{L}$-harmonic functions are locally C^∞. In particular, (3.22) gives for the choice $k = d+1$

$$\sup_{B_\rho(x)} |u| \le c\|u\|_{H^{d+1}(B_\rho(x))} \le \tilde{c}(\rho,r)\|u\|_{L^2(B_r(x))}.$$

Using a rescaling argument, we obtain for $x \in \Omega$ and $0 \le r \le \operatorname{dist}(x,\partial\Omega)$

$$\sup_{B_\rho(x)} |u| \le c_R r^{-d/2}\|u\|_{L^2(B_r(x))}, \quad 0 < \rho < r, \tag{3.23}$$

with $c_R > 0$ independent of ρ and r.

Asymptotic smoothness of the entries of S is a prerequisite of wavelet compression methods and kernel approximations based on interpolation. Additionally, it will be seen that the convergence of the ACA algorithm (see Sect. 3.4) can be proved for discrete integral operators with this type of kernel functions. Asymptotic smoothness is usually checked for each singularity matrix under investigation. The following lemma states that the entries of the singularity function of operators (3.17) always have this property. Its proof relies on the Caccioppoli inequality (see Lemma 3.3), which will be applied to $\Omega = \mathbb{R}^d$. We already know that $S(x-\cdot) \in C^\infty(D)$ is $\mathscr{L}$-harmonic in each domain $D \subset \mathbb{R}^d \setminus \{x\}$.

Lemma 3.5. *The entries of the singularity matrix $S(x-y)$ of $\mathscr{L}$ are asymptotically smooth in $\mathbb{R}^d$ with respect to y.*

Proof. Let $x \in \mathbb{R}^d$ be arbitrary but fixed. For $y \in \mathbb{R}^d \setminus \{x\}$ let $R = \|x-y\|/2$. Assume that a function u is $\mathscr{L}$-harmonic in $B_r(y)$, $0 < r < R$. Choosing $0 < \rho < r$ and $\rho' := (r+\rho)/2$, we obtain from (3.23) and Lemma 3.3 that

$$\sup_{z\in B_\rho(y)} |\partial_{z_i} u(z)|^2 \le \frac{c_R^2}{\rho'^d} \int_{B_{\rho'}(y)} |\partial_{z_i} u(z)|^2 \,\mathrm{d}z \le \frac{c_R^2 c_{\mathscr{L}}^2}{\rho'^d (r-\rho')^2} \int_{B_r(y)} |u(z)|^2 \,\mathrm{d}z \tag{3.24a}$$

$$\le 2^{d+2} \frac{\omega_d r^d}{(r+\rho)^d} \frac{c_R^2 c_{\mathscr{L}}^2}{(r-\rho)^2} \sup_{z\in B_r(y)} |u(z)|^2. \tag{3.24b}$$

Here, ω_d denotes the volume of the unit ball in $\mathbb{R}^d$.

Let $\alpha \in \mathbb{N}_0^d$ be a multi-index and $p = |\alpha|$. We define a nested sequence of balls

$$B_k = \{z \in \mathbb{R}^d : \|z-y\| < Rk/(p+1)\}, \quad k = 1,\dots,p+1,$$

centered at y. Then $B_k \subset B_R(y) \subset \mathbb{R}^d \setminus \{x\}$ and $\operatorname{dist}(B_k, \partial B_{k+1}) = R/(p+1)$. Estimate (3.24) for $\rho = Rk/(p+1)$ and $r = R(k+1)/(p+1)$ reads

$$\begin{aligned}\sup_{z\in B_k} |\partial_{z_i} u(z)| &\le 2^{d/2+1} \left(\frac{k+1}{2k+1}\right)^{d/2} \frac{c_R c_{\mathscr{L}} \sqrt{\omega_d}}{R}(p+1) \sup_{z\in B_{k+1}} |u(z)| \\ &= c'_{\mathscr{L}} \frac{p+1}{R} \left(\frac{2k+2}{2k+1}\right)^{d/2} \sup_{z\in B_{k+1}} |u(z)|, \quad k = 1,\dots,p,\end{aligned}$$

where $c'_{\mathscr{L}} := 2c_R c_{\mathscr{L}} \sqrt{\omega_d}$. Applying the previous estimate consecutively to the p partial derivatives of the function $S_{ij}(x,\cdot)$, which together with each of its derivatives $\partial_y^\alpha S_{ij}(x-y)$ is $\mathscr{L}$-harmonic in $B_R(y) \subset \mathbb{R}^d$ for arbitrary $\alpha \in \mathbb{N}_0^d$, we end up with

$$\sup_{z\in B_1} |\partial_z^\alpha S_{ij}(x-z)| \le (p+1)^{d/4} \left(\frac{c'_{\mathscr{L}}(p+1)}{R}\right)^p \sup_{z\in B_{p+1}} |S_{ij}(x-z)|$$

since $\prod_{k=1}^p \frac{2k+2}{2k+1} \le \sqrt{p+1}$. Using $(p+1)^p \le e p^p$ and Stirling's approximation

$$\sqrt{2\pi p}\left(\frac{p}{e}\right)^p < p!,$$

we obtain

$$|\partial_y^\alpha S_{ij}(x-y)| \le \frac{e}{\sqrt{2\pi}} 2^{d/4} p! \left(2^{\frac{d-2}{8}} \frac{c'_{\mathscr{L}} e}{R}\right)^p \sup_{z\in B_R(y)} |S_{ij}(x-z)|$$

due to $p \le 2^{p/2}$. Since $S_{ij}(x-\cdot)$ is harmonic on $B_R(y)$, Harnack's inequality gives $\sup_{z\in B_R(y)} |S_{ij}(x-z)| \le c_H |S_{ij}(x-y)|$ and hence the assertion. □

The previous lemma guarantees that the singularity function S of any elliptic operator is asymptotically smooth. As a consequence, the kernel function of the double-layer operator

$$\kappa(x,y) := \partial_{\nu_x} S(x-y) = \frac{1}{4\pi} \partial_{\nu_x} \|x-y\|^{-1} = \frac{1}{4\pi} \frac{\nu_x \cdot (y-x)}{\|x-y\|^3}$$

with the unit outer normal ν_x in $x \in \Gamma$ of the Laplacian in $\mathbb{R}^3$ is asymptotically smooth on Γ with respect to y. The hypersingular operator contains normal derivatives with respect to both variables such that the associated kernel function cannot be expected to be asymptotically smooth with respect to y if the boundary Γ is non-smooth. The action of the hypersingular operator, however, is usually reduced to the single- and double-layer operators via integration by parts; cf. [180, 190] in the case of the Laplacian and [167, 141] for the Lamé equations; see (3.77). Using such alternative representations, the kernel of the hypersingular operator can be deemed to be asymptotically smooth.

Example 3.6. In [52] the singularity function

$$S(x) := \frac{e^{-c\|x\|}}{\|x\|}, \quad \operatorname{Re} c > 0,$$

of the operator $-\Delta + c^2$, $c \in \mathbb{C}$, which appears, for instance, in **eddy current simulations**, is shown to be asymptotically smooth with

$$|\partial_y^\alpha S(x-y)| \le \frac{p!(\sqrt{1+\tan^2\beta}+2)^p}{\|x-y\|^{p+1}}, \quad p = |\alpha|. \tag{3.25}$$

Here, β is defined by $\tan\beta := \frac{\operatorname{Im} c}{\operatorname{Re} c}$.

The singularity function

$$S(x) := \frac{e^{\mathrm{i}\omega\|x\|}}{\|x\|}, \quad \omega \in \mathbb{R}, \quad \mathrm{i} := \sqrt{-1},$$

of the Helmholtz operator $-\Delta - \omega^2$, i.e., the limit $\operatorname{Re} c \to 0$, is not covered by this estimate since the constant in (3.25) becomes unbounded. Its asymptotic smoothness, however, can be seen from Lemma 3.5. Notice that the constant γ grows linearly with the wave number ω. Due to Remark 3.4, the non-negative constant ω^2 in the **Yukawa operator** $-\Delta + \omega^2$ does not enter γ. This observation reflects the fact that the positive term ω^2 shifts the non-negative spectrum of $-\Delta$ to the positive, while $-\omega^2$ leads to oscillations of the solution.

3.2.1 The Biharmonic Equation

A Caccioppoli-type inequality can be derived for many other problems (see [31] for regularizations of the curl-curl operator) including nonlinear ones and differential operators of higher order such as the **biharmonic operator** Δ^2. The biharmonic equation

$$\Delta^2 u = 0$$

is used to model the deflections arising in two-dimensional rectangular orthotropic symmetric laminate plates. The plate can be subjected to external perpendicular force and one is interested in the resulting deflections. Various boundary conditions can be applied to the problem.

Lemma 3.7. *Assume $u \in H^2(\Omega)$ such that $\Delta^2 u = 0$ in $D \cap \Omega$ with $u = 0$ and $\nabla u = 0$ on $D \setminus \Omega$. Then for any compact set $K \subset D$ it holds that*

$$\|\nabla u\|^2_{L^2(K)} + \sigma^2 \|\Delta u\|^2_{L^2(K)} \le \frac{c}{\sigma^2} \|u\|^2_{L^2(D)},$$

where $\sigma = \operatorname{dist}(K, \partial D)$.

Proof. Let $\eta \in C^2(D)$ satisfy $0 \le \eta \le 1$, $\eta = 1$ in K, $\eta = 0$ in a neighborhood of ∂D and $\|\nabla^p \eta\| \le c/\sigma^p$ for $0 \le p \le 3$. With $u \in H^2(\Omega)$ it follows that $\eta^2 u \in H^2_0(D \cap \Omega)$.

From

$$|\Delta(\eta^2 u)|^2 = \Delta u \Delta(\eta^4 u) + u^2 |\Delta \eta^2|^2 + 4u\Delta\eta^2 \nabla u \cdot \nabla \eta^2 + 4|\nabla u \cdot \nabla \eta^2|^2 - 2u\Delta u \|\nabla \eta^2\|^2$$

and

$$\frac{1}{4}|\nabla u \cdot \nabla \eta^2|^2 = \operatorname{div}[u\nabla\eta(\nabla\eta \cdot \nabla(\eta^2 u))] - \|\nabla\eta\|^2 u \nabla u \cdot \nabla \eta^2 - u\|\nabla\eta\|^2 \Delta(\eta^2 u) - 2u\Delta\eta\nabla(\eta^2 u)\cdot\nabla\eta$$

it follows that

$$\int_D |\Delta(\eta^2 u)|^2 \,\mathrm{d}x = \|u\Delta\eta^2\|_{L^2}^2 + \int_D (4\Delta\eta^2 - 16\|\nabla\eta\|^2) u\nabla u \cdot \nabla\eta^2 - 2\int_D u\Delta u \|\nabla\eta^2\|^2 \,\mathrm{d}x - 16\int_D u\|\nabla\eta\|^2 \Delta(\eta^2 u)\,\mathrm{d}x - 32\int_D u\Delta\eta\nabla(\eta^2 u)\cdot\nabla\eta\,\mathrm{d}x$$

due to $\int_D \Delta u \Delta(\eta^4 u)\,\mathrm{d}x = 0$ and due to the vanishing of the integrals over the divergence term. Furthermore,

$$u\Delta u\|\nabla\eta^2\|^2 = \operatorname{div}[u\|\nabla\eta^2\|^2\nabla u] - \|\nabla\eta^2\|^2\|\nabla u\|^2 - 2u\Delta\eta^2\nabla u\cdot\nabla\eta^2$$

and

$$\frac{1}{4}\|\nabla\eta^2\|^2\|\nabla u\|^2 = \operatorname{div}[u\nabla(\eta^2 u)\|\nabla\eta\|^2] - u\|\nabla\eta\|^2\Delta(\eta^2 u) - \|\nabla\eta\|^2 u\nabla u\cdot\nabla\eta^2 - 2u\Delta\eta\nabla(\eta^2 u)\cdot\nabla\eta$$

lead to

$$\int_D |\Delta(\eta^2 u)|^2\,\mathrm{d}x = \|u\Delta\eta^2\|_{L^2}^2 + 8\int_D (\Delta\eta^2 - 3\|\nabla\eta\|^2) u\nabla u\cdot\nabla\eta^2 - 24\int_D u\|\nabla\eta\|^2\Delta(\eta^2 u)\,\mathrm{d}x - 48\int_D u\Delta\eta\nabla(\eta^2 u)\cdot\nabla\eta\,\mathrm{d}x.$$

As the last step, we use the identity $2u\nabla u\cdot v = \operatorname{div}[|u|^2 v] - |u|^2\nabla v$ for all $v\in\mathbb{R}^d$, which gives

$$\int_D |\Delta(\eta^2 u)|^2\,\mathrm{d}x = \|u\Delta\eta^2\|_{L^2}^2 - 4\int_D \nabla[(2\eta\Delta\eta - \|\nabla\eta\|^2)\nabla\eta^2]|u|^2\,\mathrm{d}x - 24\int_D u\|\nabla\eta\|^2\Delta(\eta^2 u)\,\mathrm{d}x - 48\int_D u\Delta\eta\nabla(\eta^2 u)\cdot\nabla\eta\,\mathrm{d}x.$$

Due to the Cauchy-Schwarz inequality together with

$$\|u\|\nabla\eta\|^2\|_{L^2}\|\Delta(\eta^2 u)\|_{L^2} \le \frac{1}{\varepsilon}\|u\|\nabla\eta\|^2\|_{L^2}^2 + \varepsilon\|\Delta(\eta^2 u)\|_{L^2}^2$$
$$\le \frac{c^4}{\varepsilon\sigma^4}\|u\|_{L^2}^2 + \varepsilon\|\Delta(\eta^2 u)\|_{L^2}^2$$

and

$$\|u\Delta\eta\|_{L^2}\|\nabla(\eta^2 u)\cdot\nabla\eta\|_{L^2} \le \frac{1}{\varepsilon}\|u\Delta\eta\|_{L^2}^2 + \varepsilon\|\nabla(\eta^2 u)\cdot\nabla\eta\|_{L^2}^2$$
$$\le \frac{c^2}{\varepsilon\sigma^4}\|u\|_{L^2}^2 + \frac{c^2\varepsilon}{\sigma^2}\|\nabla(\eta^2 u)\|_{L^2}^2$$

for arbitrary $\varepsilon > 0$ we have

$$\int_D |\Delta(\eta^2 u)|^2\,\mathrm{d}x \le \frac{c'}{\sigma^4}\|u\|_{L^2}^2 + 24\varepsilon\|\Delta(\eta^2 u)\|_{L^2}^2 + 48\varepsilon\frac{c^2}{\sigma^2}\|\nabla(\eta^2 u)\|_{L^2}^2,$$

where c' depends on c and is bounded for $\varepsilon \to 0$. Using the Poincaré inequality $\|\nabla(\eta^2 u)\|_{L^2} \le c_P/\sigma\|\Delta(\eta^2 u)\|_{L^2}$, we arrive at

$$\left[1 - 24\varepsilon\left(1 + 2\frac{c_P^2 c^2}{\sigma^4}\right)\right]\int_D |\Delta(\eta^2 u)|^2\,\mathrm{d}x \le \frac{c'}{\sigma^4}\|u\|_{L^2}^2.$$

With the choice

$$\varepsilon = \frac{1}{48}\left(1 + 2\frac{c_P^2 c^2}{\sigma^4}\right)^{-1}$$

the assertion follows from

$$\frac{1}{\sigma^2}\|\nabla u\|_{L^2(K)}^2 + \|\Delta u\|_{L^2(K)}^2 \le \frac{1}{\sigma^2}\|\nabla(\eta^2 u)\|_{L^2(D)}^2 + \|\Delta(\eta^2 u)\|_{L^2(D)}^2$$
$$\le (c_P^2 + 1)\|\Delta(\eta^2 u)\|_{L^2(D)}^2 \le \frac{2(c_P^2+1)c'}{\sigma^4}\|u\|_{L^2(D)}^2.$$

□

The asymptotic smoothness of the arising kernel functions will be used in the following section to show existence of degenerate kernel approximations.

3.3 Approximation by Degenerate Kernels

In this section we consider matrices $A \in \mathbb{R}^{I\times J}$ with sub-blocks A_{ts} of the form

$$A_{ts} = \Lambda_{1,t}\mathscr{A}\Lambda_{2,s}^*, \tag{3.26}$$

which arise (cf. Sect. 3.1) from the discretization of integral operators

$$(\mathscr{A}v)(y) = \int_\Omega \kappa(x,y)v(x)\,\mathrm{d}\mu_x, \quad y \in \Omega. \tag{3.27}$$

Here, $\Omega \subset \mathbb{R}^d$ denotes the domain of integration and μ is an associated measure. If Ω is a $(d-1)$-dimensional manifold in $\mathbb{R}^d$, for instance, then μ denotes the surface measure. Note that strongly singular kernels are not excluded. However, then the integral in (3.27) has to be defined by an appropriate regularization.

The aim of this section is to show that the kernel function κ of $\mathscr{A}$ can be approximated under reasonable assumptions, such as the asymptotic smoothness, by a small sum of functions with separated variables; see the introductory Example 1.12.

Definition 3.8. Let $D_1, D_2 \subset \mathbb{R}^d$ be two domains. A kernel function $\kappa : D_1 \times D_2 \to \mathbb{R}$ is called **degenerate** if $k \in \mathbb{N}$ and functions $u_\ell : D_1 \to \mathbb{R}$ and $v_\ell : D_2 \to \mathbb{R}$, $\ell = 1, \ldots, k$, exist such that

$$\kappa(x,y) = \sum_{\ell=1}^{k} u_\ell(x)v_\ell(y), \quad x \in D_1, y \in D_2.$$

The number k is called **degree of degeneracy**.

Before we turn to the construction of degenerate approximants, we point out their importance for the existence of low-rank approximants to the matrix A. Due to the representation (3.26) with localizers $\Lambda_{1,t}$ and $\Lambda_{2,s}$, for a sub-block $t \times s$ of the discrete operator A the kernel function κ is evaluated only on the product of the supports X_s and Y_t of $\Lambda_{2,s}$ and $\Lambda_{1,t}$. Assume that κ is degenerate on $X_s \times Y_t$. Later on, we will find conditions on t and s for this assumption to hold. Let

$$a_\ell = \Lambda_{1,t}v_\ell \in \mathbb{R}^t \quad \text{and} \quad b_\ell = \Lambda_{2,s}u_\ell \in \mathbb{R}^s, \quad \ell = 1, \ldots, k.$$

For $z \in \mathbb{R}^s$ we have

$$b_\ell^T z = (u_\ell, \Lambda_{2,s}^* z)_{L^2(X_s)}.$$

Since for $y \in Y_t$

$$\begin{aligned}(\mathscr{A}\Lambda_{2,s}^* z)(y) &= \int_{X_s} \kappa(x,y)(\Lambda_{2,s}^* z)(x)\,\mathrm{d}\mu_x = \int_{X_s} \sum_{\ell=1}^{k} u_\ell(x)v_\ell(y)(\Lambda_{2,s}^* z)(x)\,\mathrm{d}\mu_x \\ &= \sum_{\ell=1}^{k} v_\ell(y) \int_{X_s} u_\ell(x)(\Lambda_{2,s}^* z)(x)\,\mathrm{d}\mu_x = \sum_{\ell=1}^{k} v_\ell(y) b_\ell^T z,\end{aligned}$$

we obtain for the sub-block A_{ts} of A

$$A_{ts} = \Lambda_{1,t}\mathscr{A}\Lambda_{2,s}^* = \Lambda_{1,t}\sum_{\ell=1}^{k} v_\ell b_\ell^T = \sum_{\ell=1}^{k} (\Lambda_{1,t}v_\ell) b_\ell^T = \sum_{\ell=1}^{k} a_\ell b_\ell^T. \tag{3.28}$$

The rank of A_{ts} is obviously bounded by k. Therefore, degenerate kernels lead to low-rank matrices if t and s are large enough compared with k; i.e., if $k(|t|+|s|) < |t||s|$. Note that the contrary, i.e., that a small rank k implies a degenerate kernel function of degree k, is not true in general. A block's rank is also affected by the geometry and by the discretization.

Example 3.9. We consider functions $f(x,y)$ which for each x are polynomial in y of order at most k, i.e., $f(x,\cdot) \in \Pi_k^d$. Then, f is degenerate on $\mathbb{R}^d \times \mathbb{R}^d$ of degree

$$\dim \Pi_k^d = |\left\{\alpha \in \mathbb{N}_0^d : |\alpha| \le k\right\}| = \sum_{\ell=0}^{k} \binom{\ell+d-1}{\ell} = \binom{k+d}{d} \le (k+1)^d.$$

The restriction $f(x,\cdot)|_H$ of $f(x,\cdot)$ to the hyperplane $H := \{(x,0) : x \in \mathbb{R}^{d-1}\}$ is degenerate of degree at most $(k+1)^{d-1}$ since $f(x,\cdot)|_H$ is a $(d-1)$-variate polynomial of degree at most k. The boundary $\Gamma \subset \mathbb{R}^d$ arising from practical applications often consists of hyperplanes, on which a lower degree of degeneracy thus is sufficient. Note that a reduction of the degree cannot be expected for any hypersurface.

In most applications, the kernel function κ is not degenerate. Besides, a degenerate kernel function would imply the compactness of the operator $\mathscr{A}$. The degeneracy of κ cannot even be expected on subdomains. However, many kernel functions including asymptotically smooth kernels can be approximated locally by degenerate ones.

As a starting point we present degenerate approximations of kernels together with the corresponding error estimates for problems that are usually treated by boundary integral methods. In the following three examples we assume that $x, y \in \mathbb{R}^d$ with $\|x\| > \|y\|$. Furthermore, we will use the notation $\hat{x} = x/\|x\|$ for $x \neq 0$.

Example 3.10 (Coulomb potential). The singularity function of the Laplacian $-\Delta$ is the **Coulomb potential** shown in (3.8). For $d = 3$ one can easily prove the error estimate

$$\left| \frac{1}{\|x-y\|} - \kappa_p(x,y) \right| \le \frac{1}{\|x\| - \|y\|} \left(\frac{\|y\|}{\|x\|} \right)^p, \tag{3.29}$$

where

$$\kappa_p(x,y) = \sum_{\ell=0}^{p-1} \frac{1}{2\ell+1} \frac{\|y\|^\ell}{\|x\|^{\ell+1}} P_\ell(\hat{x} \cdot \hat{y})$$

is the **multipole expansion**. The degeneracy becomes visible by the addition theorem of spherical harmonics

$$P_\ell(\hat{x} \cdot \hat{y}) = \frac{4\pi}{2\ell+1} \sum_{|m| \le \ell} Y_\ell^m(\hat{x}) Y_\ell^{-m}(\hat{y}) \tag{3.30}$$

with the spherical harmonics Y_ℓ^m of order ℓ and degree m, $|m| \le \ell$, and the ℓth Legendre polynomial P_ℓ.

Example 3.11 (Helmholtz kernel). The singularity function of the Helmholtz operator $-\Delta - \omega^2$, $\omega \in \mathbb{R}$, in d spatial dimensions is

$$S(x) = \frac{\mathrm{i}}{4} \left(\frac{\omega}{2\pi\|x\|} \right)^{d/2-1} H_{d/2-1}^{(1)}(\omega\|x\|),$$

where $H_\ell^{(1)}$ denotes the ℓth Hankel function of the first kind. The most important cases are

$$S(x) = \begin{cases} \frac{\mathrm{i}}{4} H_0^{(1)}(\omega\|x\|), & d = 2, \\ \frac{1}{4\pi} \frac{e^{\mathrm{i}\omega\|x\|}}{\|x\|}, & d = 3. \end{cases}$$

The next two error estimates are taken from [109]. We first consider the two-dimensional case. For $d = 2$ it holds that

$$\left| H_0^{(1)}(\omega\|x-y\|) - \kappa_p(x,y) \right| \leq c \, \frac{e^{\omega\|x\|}}{\sqrt{\omega\|x\|}} \, p \left(\frac{\|y\|}{\|x\|} \right)^p, \tag{3.31}$$

where

$$\kappa_p(x,y) = \sum_{\ell=0}^{p-1} H_\ell^{(1)}(\omega\|x\|) \, J_\ell(\omega\|y\|) \, T_\ell(\hat{x}\cdot\hat{y}).$$

Here, J_ℓ and T_ℓ denote the Bessel function and the Chebyshev polynomial of ℓth order. The separation of the variables x and y can be achieved by taking into account that T_ℓ is a polynomial of order ℓ.

A similar estimate is valid in three spatial dimensions:

$$\left| \frac{e^{\mathrm{i}\omega\|x-y\|}}{\|x-y\|} - \kappa_p(x,y) \right| \leq c \, \frac{e^{\omega\|x\|}}{\|x\|} \, p^2 \left(\frac{\|y\|}{\|x\|} \right)^p, \tag{3.32}$$

where

$$\kappa_p(x,y) = \mathrm{i}\pi \sum_{\ell=0}^{p-1} \left(\ell + \frac{1}{2} \right) \frac{H_{\ell+\frac{1}{2}}^{(1)}(\omega\|x\|)}{\sqrt{\|x\|}} \, \frac{J_{\ell+\frac{1}{2}}(\omega\|y\|)}{\sqrt{\|y\|}} \, P_\ell(\hat{x}\cdot\hat{y}).$$

The degeneracy follows from (3.30). Compared with the approximation of the kernel function of the Laplacian, the error estimates suffer from possibly high wave numbers ω.

Example 3.12 (Lamé kernel). The singularity function of the operator associated with the Lamé equations (3.4) is the $d \times d$, $d \geq 2$, matrix

$$S(x) = \frac{\lambda + 3\mu}{2\mu(\lambda + 2\mu)} \left[S_\Delta(x) I_d + \frac{\lambda + \mu}{\omega_d(\lambda + 3\mu)} \frac{xx^T}{\|x\|^d} \right],$$

where S_Δ is the singularity function of the Laplacian, $I_d \in \mathbb{R}^{d\times d}$ is the identity and ω_d denotes the volume of the unit sphere in $\mathbb{R}^d$. Since S is closely related with S_Δ, techniques from the approximation of S_Δ can be used to approximate S; cf. [143].

Similar techniques can be applied to the singularity function of the curl-curl operator.

Example 3.13 (curl-curl operator). The singularity function of the curl-curl operator (3.5) is the $d \times d$, $d \geq 2$, matrix

$$S(x) = \frac{\alpha+1}{2\alpha} S_\Delta(x) I_d + \frac{\alpha-1}{8\pi\alpha} \frac{xx^T}{\|x\|^d},$$

where S_Δ again denotes the singularity function of the Laplacian.

The previous four examples show that the respective kernel can be approximated at points x and y satisfying $\|x\| > \|y\|$. Let ξ_{D_2} be the **Chebyshev center** of a set $D_2 \subset \mathbb{R}^d$, i.e., the center of the ball with minimum radius ρ_{D_2} containing D_2. In order to guarantee that the latter condition holds for all x stemming from a set $D_1 \subset \mathbb{R}^d$ in the coordinate system with origin ξ_{D_2}, we have to require

$$\eta \operatorname{dist}(\xi_{D_2}, D_1) \geq \rho_{D_2}, \tag{3.33}$$

where we have added a parameter $0 < \eta < 1$ in order to obtain uniform convergence of the order η^p in the error estimates (3.29), (3.31), and (3.32).

If the matrix block A_{ts} is to be approximated by a low-rank matrix, the following condition on $t \times s$

$$\eta \operatorname{dist}(\xi_{Y_t}, X_s) \geq \rho_{Y_t} \tag{3.34}$$

has to be imposed. Condition (3.34) satisfies the requirements of an admissibility condition and can hence be used for generating an admissible block structure as described in Chap. 1. Using the degenerate kernel approximant on $X_s \times Y_t$, a low-rank matrix approximant is defined by (3.28). This principle is always exploited by methods approximating the kernel. Degenerate kernel approximations are constructed using analytic arguments and then are related to low-rank matrices by the above technique. As we have seen in Example 3.9, it seems more natural to generate the low-rank approximant directly from the matrix entries. An algorithm for this purpose will be presented in Sect. 3.4. The following approximation results, however, will be important for the convergence analysis of the method from Sect. 3.4.

The admissibility condition (3.34) does not depend on the respective kernel. In fact it will be seen in moment that it guarantees exponential convergence for all kernels stemming from elliptic problems. However, the contrary is not true. Exponentially convergent kernel approximants may exist although condition (3.34) is not satisfied; see, for instance, the so-called **weak admissibility condition** from [134].

The following example, which arises from the **Gauß transform**

$$(\mathscr{G}v)(x) = \int_{\mathbb{R}^d} e^{-\|x-y\|^2/\sigma} v(y)\,\mathrm{d}y, \quad \sigma > 0,$$

shows that also other admissibility conditions may be required. Notice that in contrast to the previous examples the kernel function of $\mathscr{G}$ is not singular at all.

Example 3.14 (Gauß kernel). The following example is not connected with any elliptic differential operator. However, it can be shown that for all $\|y\|_\infty \leq r/\sqrt{2}$, $r < 1$, it holds that (see [16])

$$|e^{-\|x-y\|^2} - \kappa_p(x,y)| \leq (1-r)^{-d} \sum_{\ell=0}^{d-1} \binom{d}{\ell} (1-r^p)^\ell \left(\frac{r^p}{\sqrt{p!}}\right)^{d-\ell},$$

where

$$\kappa_p(x,y) = \sum_{\|\alpha\|_\infty < p} \frac{1}{\alpha!} e^{-\|x\|^2} \prod_{i=1}^{d} H_{\alpha_i}(x) y^{\alpha_i}$$

and H_k is the kth Hermite polynomial. The latter expansion is used in the **fast Gauß transform**; cf. [119, 120].

The above error estimate shows that

$$|e^{-\|x-y\|^2} - \kappa_p(x,y)| \le c\,\eta^p$$

for $x \in D_1$, $y \in D_2$ provided that

$$\rho_{D_2}^{\|\cdot\|_\infty} \le \eta$$

for some parameter $0 < \eta < \sqrt{2}$.

In [208] the Taylor expansion is used for the construction of the degenerate kernel approximation.

Instead of presenting a suitable degenerate approximation for each given kernel, in the following two paragraphs we will construct degenerate kernels by approximation methods which can be equally applied to any asymptotically smooth function.

3.3.1 Degenerate Kernels through Taylor Expansion

Let $\kappa : D_1 \times D_2 \to \mathbb{R}$ be analytic with respect to its second argument y and let ξ_{D_2} denote the Chebyshev center of D_2. Then κ has a Taylor expansion

$$\kappa(x,y) = \sum_{|\alpha|<p} \frac{1}{\alpha!} \partial_y^\alpha \kappa(x,\xi_{D_2})(y-\xi_{D_2})^\alpha + R_p(x,y),$$

where

$$R_p(x,y) := \sum_{|\alpha|\ge p} \frac{1}{\alpha!} \partial_y^\alpha \kappa(x,\xi_{D_2})(y-\xi_{D_2})^\alpha$$

denotes the remainder of the expansion, which converges to zero for $p \to \infty$. The rate of convergence, however, can be arbitrarily bad. Note that

$$T_p[\kappa](x,y) := \sum_{|\alpha|<p} \frac{1}{\alpha!} \partial_y^\alpha \kappa(x,\xi_{D_2})(y-\xi_{D_2})^\alpha$$

is a degenerate kernel approximation. Since $T_p[\kappa](x,\cdot) \in \Pi_{p-1}^d$, the degree of degeneracy is the dimension of the space of d-variate polynomials of order at most $p-1$

$$k = \dim \Pi_{p-1}^d \le p^d;$$

see Example 3.9. In order to be able to guarantee a specific complexity, i.e., to specify how p depends on the accuracy ε of the approximation, we have to make sure that the expansion converges fast enough. An exponential convergence is desirable and would lead to a logarithmic dependence of p on ε.

In the rest of this section we assume that the kernel function of the integral operator (3.27) is asymptotically smooth with respect to y; see Definition 3.2. From Remark 3.1 and Lemma 3.5 we know that the kernel functions of boundary integral formulations of elliptic boundary value problems possess this property. The importance of asymptotic smoothness is that it leads to exponential convergence of the Taylor series if D_1 and D_2 are far enough away from each other.

Lemma 3.15. *Assume that* (3.33) *holds with* $\eta > 0$ *satisfying* $\gamma\sqrt{d}\eta < 1$. *If* κ *is asymptotically smooth on the convex set* D_2 *with respect to y, then it holds that*

$$|\kappa(x,y) - T_p[\kappa](x,y)| \leq c\,\frac{(\gamma\sqrt{d}\eta)^p}{1-\gamma\sqrt{d}\eta}|\kappa(x,\xi_{D_2})|$$

for all $x \in D_1$ *and* $y \in D_2$. *Here,* ξ_{D_2} *denotes the Chebyshev center of* D_2.

Proof. For the remainder R_p it holds that

$$\begin{aligned}
|R_p(x,y)| &\leq \sum_{|\alpha|\geq p} \frac{1}{\alpha!}|\partial_y^\alpha \kappa(x,\xi_{D_2})||(y-\xi_{D_2})^\alpha| \\
&\leq c\,|\kappa(x,\xi_{D_2})| \sum_{|\alpha|\geq p} \frac{\gamma^{|\alpha|}|\alpha|!}{\alpha!\|x-\xi_{D_2}\|^{|\alpha|}}|(y-\xi_{D_2})^\alpha| \\
&= c\,|\kappa(x,\xi_{D_2})| \sum_{\ell=p}^{\infty} \left(\frac{\gamma}{\|x-\xi_{D_2}\|}\right)^\ell \sum_{|\alpha|=\ell} \binom{\ell}{\alpha} |(y-\xi_{D_2})^\alpha| \\
&\leq c\,|\kappa(x,\xi_{D_2})| \sum_{\ell=p}^{\infty} \left(\gamma\sqrt{d}\frac{\|y-\xi_{D_2}\|}{\|x-\xi_{D_2}\|}\right)^\ell \\
&\leq c\,|\kappa(x,\xi_{D_2})| \sum_{\ell=p}^{\infty} (\gamma\sqrt{d}\eta)^\ell \\
&\leq c\,\frac{(\gamma\sqrt{d}\eta)^p}{1-\gamma\sqrt{d}\eta}|\kappa(x,\xi_{D_2})|
\end{aligned}$$

due to $\sum_{|\alpha|=\ell}\binom{\ell}{\alpha}|\xi^\alpha| = (\sum_{i=1}^d |\xi_i|)^\ell \leq d^{\ell/2}\|\xi\|^\ell$ for all $\xi \in \mathbb{R}^d$. □

The previous lemma shows that the Taylor expansion of asymptotically smooth kernels converges exponentially with convergence rate $\gamma\sqrt{d}\eta < 1$. Thus, $p \sim |\log\varepsilon|$ is required to achieve a given approximation accuracy $\varepsilon > 0$. For the degree k of degeneracy of $T_p[\kappa]$ it follows that

$$k \sim p^d \sim |\log\varepsilon|^d.$$

Notice that for the asymptotically smooth Coulomb potential $\kappa(x,y) := \|x-y\|^{-1}$ a degeneracy $k \sim |\log \varepsilon|^3$ is required if $d = 3$. If the multipole expansion from Example 3.10 is used instead, $k \sim |\log \varepsilon|^2$ will be enough to guarantee an error of the same order ε. Hence, the quality of the Taylor expansion, i.e., the quality of algebraic polynomials, is worse than the multipole expansion for the Coulomb potential.

Note that if κ is asymptotically smooth only with respect to the first argument x, then ρ_{D_2} has to be replaced by ρ_{D_1} in (3.33). If κ is asymptotically smooth with respect to both variables, then the symmetric condition

$$\min\{\rho_{D_1}, \rho_{D_2}\} \leq \eta \operatorname{dist}(D_1, D_2) \tag{3.35}$$

is sufficient.

3.3.2 Degenerate Kernels through Interpolation

In the preceding section we have seen that asymptotically smooth kernels can be approximated on a pair of domains satisfying (3.33). Since truncated Taylor expansions involve the computation of derivatives, this way of constructing degenerate kernel approximations has only theoretical meaning. For practical purposes the construction should rely on other approximations such as interpolation, which requires only the evaluation of the kernel function. In addition, interpolation will provide us with refined error estimates.

3.3.2.1 Interpolation on Tensor Product Grids

In this paragraph we concentrate on interpolation nodes on a tensor product grid

$$y_\alpha = (y_{\alpha_1}^{(1)}, \ldots, y_{\alpha_d}^{(d)}) \in D_2, \quad \alpha \in \mathbb{N}_0^d,\ \|\alpha\|_\infty < p, \tag{3.36}$$

where $y_i^{(\nu)} \in [a_\nu, b_\nu]$, $i = 0, \ldots, p-1$, are pairwise distinct for each $\nu = 1, \ldots, d$ and $D_2 := \prod_{\nu=1}^d [a_\nu, b_\nu]$.

We first consider the approximation of univariate functions by polynomials. In the following lemma, which is due to Melenk (cf. [184]), the approximation error $f - q$, $q \in \Pi_{p-1}$, is estimated with respect to the maximum norm $\|f\|_{\infty,M} := \sup_{x\in M} |f(x)|$.

Lemma 3.16. *Assume that $f \in C^\infty[a,b]$ is asymptotically smooth in the sense that*

$$\|f^{(p)}\|_{\infty,[a,b]} \leq c_f\, p!\, \gamma_f^p \quad \textit{for all } p \in \mathbb{N}_0.$$

Then

$$\min_{q\in\Pi_{p-1}} \|f-q\|_{\infty,[a,b]} \le 4ec_f(1+\gamma_f(b-a))p\left(1+\frac{2}{\gamma_f(b-a)}\right)^{-p}.$$

Let $y_1,\dots,y_p \in [a,b]$ be pairwise distinct points and $f \in C[a,b]$. By the previous lemma we can estimate the interpolation error $\|f-\mathfrak{I}_p f\|_{\infty,[a,b]}$ where the polynomial $\mathfrak{I}_p f \in \Pi_{p-1}$ defined by

$$(\mathfrak{I}_p f)(y) = \sum_{i=0}^{p-1} f(y_i)L_i(y), \qquad L_i(y) := \prod_{\substack{j=0\\ j\neq i}}^{p-1} \frac{y-y_j}{y_i-y_j},$$

interpolates f in y_i, $i=0,\dots,p-1$. Since $\mathfrak{I}_p$ is a linear projection onto Π_{p-1}, from

$$f-\mathfrak{I}_p f = f-q+\mathfrak{I}_p(q-f) \quad \text{for all } q\in\Pi_{p-1}$$

we obtain

$$\|f-\mathfrak{I}_p f\|_{\infty,[a,b]} \le (1+\|\mathfrak{I}_p\|) \min_{q\in\Pi_{p-1}} \|f-q\|_{\infty,[a,b]}, \tag{3.37}$$

where

$$\|\mathfrak{I}_p\| := \max\left\{\|\mathfrak{I}_p f\|_{\infty,[a,b]}/\|f\|_{\infty,[a,b]} : f\in C[a,b]\right\}$$

denotes the **Lebesgue constant**, which is invariant under affine transformations of variables. Equation (3.37) shows that approximation by the interpolation polynomial is quasi-optimal.

In the next lemma the error $\|\kappa(x,\cdot)-\mathfrak{I}_{y,p}\kappa(x,\cdot)\|_{\infty,D_2}$ of the approximation by interpolation is estimated for $d=1$.

Lemma 3.17. *Let $D_1, D_2 \subset \mathbb{R}$ such that D_2 is a closed interval and $D_1\cap D_2=\varnothing$. Let κ be asymptotically smooth with respect to y. Then for all $x\in D_1$ it holds that*

$$\|\kappa(x,\cdot)-\mathfrak{I}_{y,p}\kappa(x,\cdot)\|_{\infty,D_2} \le \bar{c}\left(1+\frac{2\operatorname{dist}(x,D_2)}{\gamma\operatorname{diam}D_2}\right)^{-p}\|\kappa(x,\cdot)\|_{\infty,D_2}, \tag{3.38}$$

$\bar{c} := 4ecp(1+\|\mathfrak{I}_p\|)\left(1+\gamma\frac{\operatorname{diam}D_2}{\operatorname{dist}(x,D_2)}\right)$.

Proof. Lemma 3.16 together with (3.37) applied to $f:=\kappa(x,\cdot)$ gives the following estimate on the interpolation error

$$\|\kappa(x,\cdot)-\mathfrak{I}_{y,p}\kappa(x,\cdot)\|_{\infty,D_2} \le (1+\|\mathfrak{I}_p\|)4ec_f(1+\gamma_f\operatorname{diam}D_2)p\left(1+\frac{2}{\gamma_f\operatorname{diam}D_2}\right)^{-p}.$$

Replacing c_f by $c\sup_{y\in D_2}|\kappa(x,y)|$ and γ_f by $\gamma/\operatorname{dist}(x,D_2)$ leads to the assertion. □

Hence, the interpolation error decays exponentially for all $x\in D_1$ if

$$\operatorname{diam}D_2 \le \eta\operatorname{dist}(D_1,D_2)$$

for *any* $\eta > 0$. Note that the well-known expression for the interpolation error

$$f(y) - (\mathfrak{I}_p f)(y) = \frac{f^{(p)}(\xi)}{p!} \prod_{i=0}^{p-1} (y - y_i)$$

for some $\xi \in \operatorname{conv}\{y, y_0, \ldots, y_{p-1}\}$ instead of Lemma 3.16 would have led to exponential convergence only if $0 < \eta < 1/\gamma$. An upper bound on η was also required by all previous constructions such as the Taylor expansion.

It remains to find a bound for $\|\mathfrak{I}_p\|$ by choosing the interpolation nodes. If the **Chebyshev nodes**

$$t_j := \frac{a+b}{2} + \frac{b-a}{2} \cdot \cos\left(\frac{2j+1}{2p}\pi\right), \quad j = 0, \ldots, p-1, \tag{3.39}$$

are used for interpolation, then the Lebesgue constant can be shown (cf. [234]) to depend only logarithmically on p

$$\|\mathfrak{I}_p\| \leq 1 + \frac{2}{\pi} \log p, \tag{3.40}$$

which is asymptotically optimal. Later in this book we will exploit that the interpolation can be rewritten in the form (see [234])

$$(\mathfrak{I}_p f)(y) = \sum_{i=0}^{p-1} c_i T_i\left(2\frac{y-a}{b-a} - 1\right), \tag{3.41}$$

where

$$c_0 := \frac{1}{p} \sum_{j=0}^{p-1} f(t_j) \quad \text{and} \quad c_i := \frac{2}{p} \sum_{j=0}^{p-1} f(t_j) \cos i\frac{2j+1}{2p}\pi, \quad i = 1, \ldots, p-1. \tag{3.42}$$

Here, $T_p(t) := \cos(p \arccos(t))$ denotes the pth Chebychev polynomial.

These properties of the univariate interpolation can be exploited for interpolating multivariate functions $f : D_2 \to \mathbb{R}$ if we use the tensor product nodes from (3.36) and the tensor product polynomials

$$\mathfrak{I}_p f := \mathfrak{I}_p^{(1)} \cdots \mathfrak{I}_p^{(d)} f \in \Pi^d_{(p-1)d}, \tag{3.43}$$

where $\mathfrak{I}_p^{(\nu)} f$ denotes the univariate interpolation operator applied to the νth argument of f. Note that $\mathfrak{I}_{y,p}\kappa$ is a degenerate kernel of degree p^d since

$$\mathfrak{I}_{y,p}\kappa(x,y) = \sum_{\|\alpha\|_\infty < p} \kappa(x, t_\alpha) L_\alpha(y),$$

where $L_\alpha(y^{(1)}, \ldots, y^{(d)}) := \prod_{\nu=1}^d L_{\alpha_\nu}(y^{(\nu)}) \in \Pi^d_{(p-1)d}$ is the product of univariate Lagrange polynomials L_{α_ν}. For the interpolation error we obtain from

$$\begin{aligned}\|f-\mathfrak{I}_p f\|_\infty &\le \|f-\mathfrak{I}_p^{(1)} f\|_\infty + \|\mathfrak{I}_p^{(1)}(f-\mathfrak{I}_p^{(2)}\cdots\mathfrak{I}_p^{(d)} f)\|_\infty \\ &\le \|f-\mathfrak{I}_p^{(1)} f\|_\infty + \|\mathfrak{I}_p^{(1)}(f-\mathfrak{I}_p^{(2)} f)\|_\infty + \dots \\ &\quad \dots + \|\mathfrak{I}_p^{(1)}\cdots\mathfrak{I}_p^{(d-1)}(f-\mathfrak{I}_p^{(d)} f)\|_\infty\end{aligned}$$

that

$$\|f-\mathfrak{I}_p f\|_\infty \le \sum_{i=1}^{d} \|f-\mathfrak{I}_p^{(i)} f\|_\infty \prod_{\nu=1}^{i-1} \|\mathfrak{I}_p^{(\nu)}\|. \tag{3.44}$$

Theorem 3.18. *Let $D_1 \subset \mathbb{R}^d$ and $D_2 = \prod_{\nu=1}^d [a_\nu, b_\nu]$ such that*

$$\eta \operatorname{dist}(D_1, D_2) \ge \max_{\nu=1,\dots,d} b_\nu - a_\nu,$$

holds for an $\eta > 0$ satisfying $c\gamma\eta < 1$. Let $\kappa(x,y)$ be asymptotically smooth with respect to y. Then for all $x \in D_1$ and $y \in D_2$ it holds that

$$|\kappa(x,y) - \mathfrak{I}_{y,p}\kappa(x,y)| \le \tilde{c}\, p \left(1+\frac{2}{\pi}\log p\right)^d \left(\frac{\gamma\eta}{2+\gamma\eta}\right)^p |\kappa(x,y)|.$$

Proof. Applying (3.44) to $f(z) := \kappa(x, y^{(1)}, \dots, y^{(\nu-1)}, z, y^{(\nu+1)}, \dots, y^{(n)})$, (3.40) and Lemma 3.17 yield for $y^{(\nu)} \in [a_\nu, b_\nu]$

$$|f(y^{(\nu)}) - \mathfrak{I}_p f(y^{(\nu)})| \le \hat{c}\, p \left(1+\frac{2}{\pi}\log p\right)^d \left(\frac{\gamma\eta}{2+\gamma\eta}\right)^p \sum_{\nu=1}^{d} \|f\|_{\infty,[a^{(\nu)},b^{(\nu)}]},$$

where $\hat{c} := 8ec(1+\gamma\eta)$. Let $z_0 \in [a^{(\nu)}, b^{(\nu)}]$ be chosen such that $|f(z_0)| = \|f\|_{\infty,[a_\nu,b_\nu]}$. Then for some $\tilde{z} \in [a_\nu, b_\nu]$ we have

$$|f(y^{(\nu)}) - f(z_0)| = |(y^{(\nu)} - z_0) f'(\tilde{z})| \le c\gamma \frac{b_\nu - a_\nu}{\|x-\tilde{y}\|} |f(\tilde{z})| \le c\gamma\eta |f(z_0)|,$$

where $\tilde{y} = (y^{(1)}, \dots, y^{(\nu-1)}, \tilde{z}, y^{(\nu+1)}, \dots, y^{(n)})$. The assertion follows from

$$\|f\|_{\infty,[a_\nu,b_\nu]} \le (1-c\gamma\eta)^{-1} |f(y^{(\nu)})| = |\kappa(x,y)|.$$

□

3.3.2.2 Interpolation on Arbitrary Nodes

The interpolation on tensor grids requires the kernel to be defined in a box. If the domain of integration Ω in (3.27) is a boundary and if its normal appears in the kernel as it does in case of the double-layer potential operator, then tensor product grids become inconvenient. In this case, we are better off using interpolation on arbitrary nodes. In addition, leaving the tensor interpolation approach behind, arbitrary nodes will lead to a reduction of the degree of degeneracy.

Let $y_\alpha \in \mathbb{R}^d$, $\alpha \in \mathbb{N}_0^d$, $|\alpha| < p$, be pairwise distinct points. In contrast to univariate interpolation, the interpolation in $\mathbb{R}^d$, $d \geq 2$, may happen to be not unique. The nodes must not lie on a hypersurface of degree $p-1$, or equivalently, there is no polynomial in Π_{p-1}^d which vanishes in all of the points. However, the set of points for which Lagrange interpolation is not unique has measure zero and the existence of interpolation polynomials is always guaranteed. Hence, we may safely assume that the interpolation is unique, i.e., that

$$W := (y_\alpha^\beta)_{\alpha,\beta} \in \mathbb{R}^{k\times k}, \quad k = \dim \Pi_{p-1}^d,$$

is invertible. Note that the uniqueness of interpolation on tensor product grids is inherited from univariate interpolation due to the special configuration of the nodes.

Using the Lagrange polynomials

$$L_\alpha(y) := \frac{\det W_\alpha(y)}{\det W} \in \Pi_{p-1}^d, \quad |\alpha| < p,$$

where $W_\alpha(y)$ denotes the matrix which arises from W by replacing row α by the vector $(y^\beta)_{|\beta|<p}$. The interpolation polynomial has the representation

$$(\mathfrak{I}_p f)(y) = \sum_{|\alpha|<p} f(y_\alpha) L_\alpha(y).$$

The following theorem is due to Sauer and Xu. It describes an upper bound for the error $E(f) := f - \mathfrak{I} f$ of multivariate Lagrangian polynomial interpolation. We use the notation of [225].

Theorem 3.19. *Let the Lagrange interpolation in the points $x^0,\dots,x^n$ be unique. For $f \in C^{n+1}(\mathbb{R}^d)$ it holds that*

$$|E_{n+1}(f)(x)| \leq \sum_{\mu\in\Lambda_n} \frac{1}{(n+1)!} |P_{\mu_n}^{[n]}(x)\pi_\mu(x^\mu)| \, \|D_{x-x_{\mu_n}^{(n)}} D_{x^\mu}^n f\|_\infty, \quad x \in \mathbb{R}^d, \quad (3.45)$$

where it suffices to take the maximum over the convex hull of $\{x^0,\dots,x^n,x\}$.

Hence, for the approximant $\kappa_p(x,\cdot) := \mathfrak{I}_p \kappa(x,\cdot)$ of κ we have

$$\kappa_p(x,y) = \sum_{|\alpha|<p} \kappa(x,y_\alpha) L_\alpha(y),$$

which is a degenerate kernel function of degree $\dim \Pi_{p-1}^d$. For $d=2$ we have that $\dim \Pi_{p-1}^d = p(p+1)/2$, whereas the degree of degeneracy of tensor product interpolations was p^2 for the same order of accuracy η^p.

3.3.3 Another Kind of Approximation

The above constructions are mainly based on the approximation of $\kappa(x,\cdot)$ by algebraic polynomials. Due to the smoothness of κ, exponential error estimates could be obtained on pairs of domains (D_1, D_2) which are far enough away from each other. However, the class of polynomials may not reflect the properties of κ in an optimal way. For instance, we have noticed that for the approximation of the Coulomb potential the multipole expansion is better suited than the Taylor expansion or polynomial interpolation.

In this section an approximation technique will be presented which is not based on any specific system of functions. Instead, restrictions of κ will be used as an approximation basis. This approach provides quasi-optimal approximation in the sense that, up to constants, the quality of the presented approximation is better than the quality of any given system $\Xi := \{\xi_1, \ldots, \xi_k\}$ of functions; i.e.,

$$E_k \leq c \inf_{\Xi} \sup_{x \in D_1} \inf_{p \in \operatorname{span} \Xi} \|\kappa(x,\cdot) - p\|_{\infty, D_2},$$

where E_k denotes the error associated with the presented approximation at a degree of degeneracy k.

To explain the principle idea of the approximation, consider the function

$$\tilde{\kappa}(x,y) := \frac{\kappa(x,y_0)\kappa(x_0,y)}{\kappa(x_0,y_0)}$$

with fixed $x_0 \in D_1$, $y_0 \in D_2$ close to x and y, respectively. The expression for $\tilde{\kappa}$ is only meaningful if $\kappa(x_0,y_0) \neq 0$. The degree of degeneracy of $\tilde{\kappa}$ is one and it holds that

$$\begin{aligned} \tilde{\kappa}(x_0,y) &= \kappa(x_0,y) \quad \text{for all } y \in D_2, \\ \tilde{\kappa}(x,y_0) &= \kappa(x,y_0) \quad \text{for all } x \in D_1. \end{aligned}$$

Hence, $\tilde{\kappa}$ interpolates κ on whole domains. Note that a degenerate kernel approximation of degree one which is based on polynomials does achieve interpolation only in a single point in general. The error $|\kappa(x,y) - \tilde{\kappa}(x,y)|$ of the approximation can be related to known error estimates by the relation

$$\begin{aligned} |\kappa(x,y) - \tilde{\kappa}(x,y)| &= |\kappa(x,y) - \frac{\kappa(x,y_0)\kappa(x_0,y)}{\kappa(x_0,y_0)}| \\ &= |\kappa(x,y) - \kappa(x,y_0) - \frac{\kappa(x,y_0)}{\kappa(x_0,y_0)}(\kappa(x_0,y) - \kappa(x_0,y_0))| \\ &\leq |\kappa(x,y) - \kappa(x,y_0)| + \frac{|\kappa(x,y_0)|}{|\kappa(x_0,y_0)|}|\kappa(x_0,y) - \kappa(x_0,y_0)|. \end{aligned}$$

Assume that $|\kappa(x,y_0)| \leq |\kappa(x_0,y_0)|$ for all $x \in D_1$, then

$$|\kappa(x,y) - \tilde{\kappa}(x,y)| \leq 2 \max_{z \in D_1} |\kappa(z,y) - \kappa(z,y_0)|.$$

The expression of the right-hand side is of the order η due to the interpolation results from the previous sections.

Since we are not satisfied with first order approximation, we will generate a sequence of degenerate kernels which will be shown to converge exponentially to κ. For convenience let

$$\kappa(x,[y]_k) = \begin{bmatrix} \kappa(x,y_{j_1}) \\ \vdots \\ \kappa(x,y_{j_k}) \end{bmatrix} \in \mathbb{R}^k \quad \text{and} \quad \kappa([x]_k,y) = \begin{bmatrix} \kappa(x_{i_1},y) \\ \vdots \\ \kappa(x_{i_k},y) \end{bmatrix} \in \mathbb{R}^k$$

with points $x_{i_\ell} \in D_1$ and $y_{j_\ell} \in D_2$, $\ell = 1,\ldots,k$. With this notation, we will construct degenerate approximations of the form

$$\kappa(x,y) = \kappa(x,[y]_k)^T W_k^{-1} \kappa([x]_k,y) + r_k(x,y), \tag{3.46}$$

where we define the $k \times k$ matrix W_k by

$$W_k = \begin{bmatrix} \kappa(x_{i_1},y_{j_1}) & \ldots & \kappa(x_{i_1},y_{j_k}) \\ \vdots & & \vdots \\ \kappa(x_{i_k},y_{j_1}) & \ldots & \kappa(x_{i_k},y_{j_k}) \end{bmatrix}.$$

These results will be used in the next section to derive an iterative algorithm for the approximation of matrices generated by low-rank matrices without knowing the rank of the approximation in advance.

Let us first turn to the analytic problem of approximating a general asymptotically smooth kernel by a degenerate kernel. In [17] sequences $\{s_k\}$, $\{r_k\}$ for the approximation of κ have been defined by the following rule

$$r_0(x,y) = \kappa(x,y), \quad s_0(x,y) = 0,$$

and for $k = 0,1,\ldots$

$$r_{k+1}(x,y) = r_k(x,y) - \gamma_{k+1}\, r_k(x,y_{j_{k+1}})\, r_k(x_{i_{k+1}},y) \tag{3.47a}$$
$$s_{k+1}(x,y) = s_k(x,y) + \gamma_{k+1}\, r_k(x,y_{j_{k+1}})\, r_k(x_{i_{k+1}},y) \tag{3.47b}$$

where $\gamma_{k+1} = \left(r_k(x_{i_{k+1}},y_{j_{k+1}})\right)^{-1}$ and $x_{i_{k+1}}$ and $y_{j_{k+1}}$ are chosen in each step so that $r_k(x_{i_{k+1}},y_{j_{k+1}}) \neq 0$. The above approximation by s_k was reinvented under the name "Geddes series expansion" in [63].

Notice that the functions r_k accumulate zeros. Thus s_k gradually interpolates κ.

Lemma 3.20. *For $1 \leq \ell \leq k$ it holds that $r_k(x,y_{j_\ell}) = 0$ for all $x \in D_1$ and $r_k(x_{i_\ell},y) = 0$ for all $y \in D_2$.*

Proof. The lemma holds for $\ell = k$ since

$$r_k(x,y_{j_k}) = r_{k-1}(x,y_{j_k}) - \gamma_k r_{k-1}(x,y_{j_k}) r_{k-1}(x_{i_k},y_{j_k}) = 0.$$

We will prove the rest by induction from $k-1$ to k. We just saw that the lemma is true for $k=1$. Assume it holds for $k-1$, then we have $r_{k-1}(x,y_{i_\ell}) = 0$ for all $x \in D_1$ and all $1 \le \ell < k$. Hence, from (3.47)

$$r_k(x,y_{j_\ell}) = r_{k-1}(x,y_{j_\ell}) - \gamma_k r_{k-1}(x,y_{j_k}) r_{k-1}(x_{i_k},y_{j_\ell}) = 0.$$

Interchanging the roles of x and y, we also obtain that $r_k(x_{i_\ell},y) = 0$ for $1 \le \ell \le k$ and all $y \in D_2$. □

Let $W_k^{(\ell)}(x) \in \mathbb{R}^{k\times k}$ be the matrix which arises from replacing the ℓth row of W_k by the vector $\kappa(x,[y]_k)$. For the determinant of $W_k^{(\ell)}(x)$ the following recursive relation can be shown.

Lemma 3.21. *For* $1 \le \ell < k$

$$\det W_k^{(\ell)}(x) = r_{k-1}(x_{i_k},y_{j_k}) \det W_{k-1}^{(\ell)}(x) - r_{k-1}(x,y_{j_k}) \det W_{k-1}^{(\ell)}(x_{i_k})$$

holds and

$$\begin{aligned} \det W_1^{(1)}(x) &= r_0(x,y_{j_1}), \\ \det W_k^{(k)}(x) &= r_{k-1}(x,y_{j_k}) \det W_{k-1}, \quad k>1. \end{aligned}$$

Especially,

$$\det W_k = r_0(x_{i_1},y_{j_1}) \cdot \ldots \cdot r_{k-1}(x_{i_k},y_{j_k}).$$

Proof. From (3.47) it is easy to see that there are coefficients $\alpha_i^{(k-1)}$, $i=1,\ldots,k-1$, so that for all $x \in D_1$

$$r_{k-1}(x,y_{j_k}) = \kappa(x,y_{j_k}) - \sum_{\nu=1}^{k-1} \alpha_\nu^{(k-1)} \kappa(x,y_{j_\nu}).$$

Thus, it is possible to replace each entry $\kappa(\cdot,y_{j_k})$ in the last column of $W_k^{(\ell)}(x)$ by $r_{k-1}(\cdot,y_{j_k})$ and obtain $\tilde{W}_k^{(\ell)}(x)$ without changing the determinant. Since from the previous lemma one has $r_{k-1}(x_{i_\nu},y_{j_k}) = 0$, $1 \le \nu < k$, only the ℓth and the kth entry of the last column of $\tilde{W}_k^{(\ell)}(x)$ do not vanish. Laplace's theorem yields the assertion. □

The previous lemma guarantees that W_k is non-singular provided that each $r_{k-1}(x_{i_k},y_{j_k}) \neq 0$. We are now able to show that the decomposition of κ into s_k and r_k is of type (3.46).

Lemma 3.22. *For the generated sequences* s_k *and* r_k, $k \ge 0$, *it holds that*

$$s_k(x,y) + r_k(x,y) = \kappa(x,y),$$

where for $k \geq 1$

$$s_k(x,y) = \kappa(x,[y]_k)^T W_k^{-1} \kappa([x]_k,y).$$

Proof. The lemma is obviously true for $k = 1$. We continue by induction. From the definition of r_k and s_k we can see that

$$s_k(x,y) + r_k(x,y) = s_{k-1}(x,y) + r_{k-1}(x,y),$$

which according to the induction coincides with $\kappa(x,y)$. For the sake of brevity we set

$$a_k = W_{k-1}^{-1}\,\kappa([x]_{k-1}, y_{j_k}) \quad \text{and} \quad b_k = W_{k-1}^{-T}\,\kappa(x_{i_k},[y]_{k-1}).$$

Since due to the induction

$$\begin{aligned} s_k(x,y) &= s_{k-1}(x,y) + \gamma_k\, r_{k-1}(x,y_{j_k}) r_{k-1}(x_{i_k},y) \\ &= \begin{bmatrix} \kappa(x,[y]_{k-1}) \\ \kappa(x,y_{j_k}) \end{bmatrix}^T \begin{bmatrix} W_{k-1}^{-1} + \gamma_k a_k b_k^T & -\gamma_k a_k \\ -\gamma_k b_k^T & \gamma_k \end{bmatrix} \begin{bmatrix} \kappa([x]_{k-1},y) \\ \kappa(x_{i_k},y) \end{bmatrix} \end{aligned}$$

and

$$W_k \begin{bmatrix} W_{k-1}^{-1} + \gamma_k a_k b_k^T & -\gamma_k a_k \\ -\gamma_k b_k^T & \gamma_k \end{bmatrix} = I_k,$$

we obtain the representation

$$s_k(x,y) = \kappa(x,[y]_k)^T W_k^{-1} \kappa([x]_k,y)$$

also for s_k. □

Remark 3.23. Using Cramer's rule we see that

$$\left(W_k^{-1}\kappa([x]_k,y)\right)_\ell = \frac{\det V_k^{(\ell)}(y)}{\det W_k}, \quad \ell = 1,\dots,k, \tag{3.48}$$

where $V_k^{(\ell)}(y)$ is the matrix which arises from W_k by replacing the ℓth column by $\kappa([x]_k,y)$. Hence, we obtain

$$s_k(x,y) = \sum_{\ell=1}^{k} \frac{\det V_k^{(\ell)}(y)}{\det W_k}\,\kappa(x,y_{j_\ell}). \tag{3.49}$$

Observe that

$$L_\ell(y) := \frac{\det V_k^{(\ell)}(y)}{\det W_k} \;\in\; \operatorname{span}\{\kappa(x_{i_\nu},y),\, \nu = 1,\dots,k\}$$

satisfies $L_\mu(y_{j_\nu}) = \delta_{\mu\nu}$, $1 \leq \mu,\nu \leq k$, and is therefore the ℓth Lagrange function for the interpolation system $\{\kappa(x_{i_\nu},y),\, \nu = 1,\dots,k\}$. The representation (3.49) hence

shows that s_k is the uniquely defined interpolant of κ at the nodes y_{j_ℓ} in the span of the functions $\kappa(x_{i_\ell}, y)$, $\ell = 1, \ldots, k$.

3.3.3.1 Error Analysis

In what follows it will be shown that the remainder r_k can be estimated by the remainder of the best approximation in any system $\Xi = \{\xi_1, \ldots, \xi_k\}$ of functions. The mentioned approximation by polynomials is an example. Further examples are the approximation by spherical harmonics or the **sinc interpolation**; cf. [247, 134]. For the uniqueness of interpolation in this system we assume that the matrix $(\xi_\mu(y_{j_\nu}))_{\mu,\nu}$ is non-singular.

Denote by

$$\|\mathfrak{I}_k^\Xi\| := \max\{\|\mathfrak{I}_k^\Xi f\|_{\infty,D_2} / \|f\|_{\infty,D_2} : f \in C(D_2)\}$$

the Lebesgue constant of the interpolation operator $\mathfrak{I}_k^\Xi$ defined by

$$\mathfrak{I}_k^\Xi f := \sum_{\ell=1}^{k} f(y_{j_\ell}) L_\ell^\Xi$$

with L_ℓ^Ξ, $\ell = 1, \ldots, k$, being the Lagrange functions for ξ_ℓ and y_{j_ℓ}, $\ell = 1, \ldots, k$. Since

$$f - \mathfrak{I}_k^\Xi f = f - p + \mathfrak{I}_k^\Xi (p - f) \quad \text{for all } p \in \operatorname{span} \Xi,$$

it follows that, up to constants, the interpolation error $E_k^\Xi(f) := f - \mathfrak{I}_k^\Xi f$ is bounded by the error of the best approximation

$$\|E_k^\Xi(f)\|_{\infty,D_2} \le (1 + \|\mathfrak{I}_k^\Xi\|) \inf_{p \in \operatorname{span} \Xi} \|f - p\|_{\infty,D_2}. \tag{3.50}$$

Defining $\kappa_x : D_2 \to \mathbb{R}$ by $\kappa_x(y) := \kappa(x, y)$ for $y \in D_2$ and fixed $x \in D_1$, it is hence enough to estimate r_k by the error $E_k^\Xi(\kappa_x)$ of the interpolation in Ξ if we want to estimate r_k by the best approximation error in Ξ.

Lemma 3.24. *For $x \in D_1$ and $y \in D_2$ it holds that*

$$r_k(x, y) = E_k^\Xi(\kappa_x)(y) - \sum_{\ell=1}^{k} \frac{\det W_k^{(\ell)}(x)}{\det W_k} E_k^\Xi(\kappa_{x_{i_\ell}})(y).$$

Proof. Let

$$L^\Xi(y) = \begin{bmatrix} L_1^\Xi(y) \\ \vdots \\ L_k^\Xi(y) \end{bmatrix}$$

be the vector of the Lagrange functions L_ℓ^Ξ, $\ell = 1, \ldots, k$, to the points $y_{j_1}, \ldots, y_{j_k}$. Using Lemma 3.22 gives

$$\begin{aligned} r_k(x,y) &= \kappa(x,y) - \kappa(x,[y]_k)^T W_k^{-1} \kappa([x]_k,y) \\ &= \kappa(x,y) - \kappa(x,[y]_k)^T L^{\Xi}(y) - \kappa(x,[y]_k)^T W_k^{-1} \left(\kappa([x]_k,y) - W_k L^{\Xi}(y)\right) \\ &= E_k^{\Xi}(\kappa_x)(y) - \sum_{\ell=1}^{k} \left(\kappa(x,[y]_k)^T W_k^{-1}\right)_\ell E_k^{\Xi}(\kappa_{x_{i_\ell}})(y). \end{aligned}$$

The assertion follows from (3.48). □

We are able to control the approximation error by making assumptions on the choice of the points $x_{i_1},\dots,x_{i_k}$. Assume that our choice of points $x_{i_1},\dots,x_{i_k}$ leads to a submatrix W_k whose determinant cannot be increased by interchanging one row by any vector $\kappa(x,[y]_k)$, $x \in D_1$, i.e.,

$$|\det W_k| \geq |\det W_k^{(\ell)}(x)|, \quad 1 \leq \ell \leq k,\ x \in D_1. \tag{3.51}$$

These matrices of maximum volume play an important role in interpolation theory; cf. [94]. From the previous lemma we obtain

$$|r_k(x,y)| \leq (k+1) \sup_{z \in \{x,x_{i_1},\dots,x_{i_k}\}} |E_k^{\Xi}(\kappa_z)(y)|. \tag{3.52}$$

Instead of choosing the points $x_{i_1},\dots,x_{i_k}$ according to condition (3.51), which is difficult to check in practice, we may choose x_{i_k} in each step so that $r_{k-1}(x_{i_k},y_{i_k})$ is the maximum element in modulus. From Lemma 3.21 we see that this is the best possible choice with respect to maximum determinants if we keep all other previously chosen elements fixed. The following lemma states that with this choice of points (3.51) can be satisfied with an additional factor.

Lemma 3.25. *Assume in each step we choose x_{i_k} so that*

$$|r_{k-1}(x_{i_k},y_{j_k})| \geq |r_{k-1}(x,y_{j_k})| \quad \textit{for all } x \in D_1.$$

Then for $1 \leq \ell \leq k$ it holds that

$$\sup_{x \in D_1} \frac{|\det W_k^{(\ell)}(x)|}{|\det W_k|} \leq 2^{k-\ell}.$$

Proof. Using Lemma 3.21 gives for $1 \leq \ell < k$

$$\frac{\det W_k^{(\ell)}(x)}{\det W_k} = \frac{\det W_{k-1}^{(\ell)}(x)}{\det W_{k-1}} - \frac{r_{k-1}(x,y_{j_k})}{r_{k-1}(x_{i_k},y_{j_k})} \frac{\det W_{k-1}^{(\ell)}(x_{i_k})}{\det W_{k-1}}$$

and

$$\frac{\det W_k^{(k)}(x)}{\det W_k} = \frac{r_{k-1}(x,y_{j_k})}{r_{k-1}(x_{i_k},y_{j_k})}.$$

Thus, we obtain for $1 \leq \ell < k$

$$\sup_{x\in D_1} \frac{|\det W_k^{(\ell)}(x)|}{|\det W_k|} \le 2 \sup_{x\in D_1} \frac{|\det W_{k-1}^{(\ell)}(x)|}{|\det W_{k-1}|}$$

from what the assertion follows. □

As a consequence of the previous lemma, instead of (3.52) we find

$$|r_k(x,y)| \le 2^k \sup_{z\in\{x,x_{i_1},\dots,x_{i_k}\}} |E_k^{\Xi}(\kappa_z)(y)| \tag{3.53a}$$

$$\le 2^k (1+\|\mathfrak{I}_k^{\Xi}\|) \sup_{z\in\{x,x_{i_1},\dots,x_{i_k}\}} \inf_{p\in\operatorname{span}\Xi} \|\kappa(z,\cdot)-p\|_{\infty,D_2}. \tag{3.53b}$$

due to (3.50). This estimate implies that the presented approximation scheme gives quasi-optimal results since, up to constants, the approximation error r_k is smaller than the approximation error associated with any system of functions $\Xi = \{\xi_1,\dots,\xi_k\}$. Note that both approximations lead to the same degree of degeneracy. We remark that the exponentially growing factor 2^k is a worst-case estimate. The same factor will appear in the *adaptive cross approximation* method from Sect. 3.4, which is the algebraic analogue of the presented approximation. There, this factor can be related to the growth factor appearing in the *LU* decomposition with partial pivoting, which is known to be hardly observable in practice.

Since we are only interested in kernel approximation as a means to construct matrix approximants, in the following section we return to the algebraic problem.

3.3.4 Matrix Approximation Error

Assume that the block $b = t \times s$ satisfies an admissibility condition, which guarantees the existence of a degenerate kernel approximation $\tilde{\kappa}$ on the Cartesian product of $D_1 = X_s$ and $D_2 = Y_t$. Using (3.28), the kernel approximant $\tilde{\kappa}$ can be used to define the low-rank approximant $\tilde{A}_{ts} = \Lambda_{1,t}\tilde{\mathscr{A}}\Lambda_{2,s}^*$ to the matrix block A_{ts} defined in (3.26). Here, $\tilde{\mathscr{A}}$ denotes the operator

$$(\tilde{\mathscr{A}}v)(y) := \int_\Omega \tilde{\kappa}(x,y)v(x)\,\mathrm{d}\mu_x.$$

The next theorem answers the question how the size of an approximation error $\kappa - \tilde{\kappa}$ propagates to the size of the matrix error $A_{ts} - \tilde{A}_{ts}$. We may confine ourselves to admissible blocks b, since non-admissible blocks are represented without approximation. In order to be able to control the influence of the discretization on the size of the matrix entries, we assume that there are constants $c_{\Lambda_1}, c_{\Lambda_2} > 0$, which usually depend on the grid size, such that

$$\|\Lambda_{1,t}u\|_2 \le c_{\Lambda_1}\|u\|_{L^2(Y_t)} \quad \text{and} \quad \|\Lambda_{2,s}v\|_2 \le c_{\Lambda_2}\|v\|_{L^2(X_s)}$$

for all $u \in L^2(Y_t)$, $v \in L^2(X_s)$.

Theorem 3.26. *Let $b = t \times s$, $t \subset I$ and $s \subset J$. Assume that*

$$|\kappa(x,y) - \tilde{\kappa}(x,y)| \leq \varepsilon \quad \text{for all } x \in X_s,\ y \in Y_t.$$

Then, in the case of Galerkin matrices

$$\begin{aligned} |a_{ij} - \tilde{a}_{ij}| &\leq \varepsilon \|\psi_i\|_{L^1} \|\varphi_j\|_{L^1}, \quad i \in t,\ j \in s, \\ \|A_b - \tilde{A}_b\|_F &\leq \varepsilon c_{\Lambda_1} c_{\Lambda_2} [\mu(X_s)\mu(Y_t)]^{1/2}, \end{aligned}$$

in the case of collocation matrices

$$\begin{aligned} |a_{ij} - \tilde{a}_{ij}| &\leq \varepsilon \|\varphi_j\|_{L^1}, \quad i \in t,\ j \in s, \\ \|A_b - \tilde{A}_b\|_F &\leq \varepsilon c_{\Lambda_2} [|t|\mu(X_s)]^{1/2} \end{aligned}$$

and in the case of Nyström matrices

$$\begin{aligned} |a_{ij} - \tilde{a}_{ij}| &\leq \varepsilon, \quad i \in t,\ j \in s, \\ \|A_b - \tilde{A}_b\|_F &\leq \sqrt{|t||s|}\,\varepsilon. \end{aligned}$$

Proof. For $i \in t$ and $j \in s$ we find for the entries of Galerkin matrices

$$\begin{aligned} |a_{ij} - \tilde{a}_{ij}| &= \left| \int_{X_s} \int_{Y_t} [\kappa(x,y) - \tilde{\kappa}(x,y)] \psi_i(y) \varphi_j(x) \, \mathrm{d}\mu_y \, \mathrm{d}\mu_x \right| \\ &\leq \int_{X_s} \int_{Y_t} |\kappa(x,y) - \tilde{\kappa}(x,y)| |\psi_i(y)| |\varphi_j(x)| \, \mathrm{d}\mu_y \, \mathrm{d}\mu_x \\ &\leq \varepsilon \|\psi_i\|_{L^1} \|\varphi_j\|_{L^1}. \end{aligned}$$

In the case of collocation methods we obtain

$$\begin{aligned} |a_{ij} - \tilde{a}_{ij}| &= \left| \int_{X_s} [\kappa(x,y_i) - \tilde{\kappa}(x,y_i)] \varphi_j(x) \, \mathrm{d}\mu_x \right| \\ &\leq \int_{X_s} |\kappa(x,y_i) - \tilde{\kappa}(x,y_i)| |\varphi_j(x)| \, \mathrm{d}\mu_x \leq \varepsilon \|\varphi_j\|_{L^1}. \end{aligned}$$

The entrywise estimate in the case of Nyström matrices is trivial.

We turn to the Frobenius norm estimates. In the case of Galerkin matrices we define the linear operator $\Lambda : L^2(X_s \times Y_t) \to \mathbb{R}^{t \times s}$ by

$$(\Lambda f)_{ij} = \int_{X_s} \int_{Y_t} f(x,y) \psi_i(y) \varphi_j(x) \, \mathrm{d}\mu_y \, \mathrm{d}\mu_x, \quad i \in t,\ j \in s$$

From

$$\begin{aligned}\|\Lambda f\|_F^2 &= \sum_{i\in t}\sum_{j\in s}\left[\int_{X_s}\left(\int_{Y_t} f(x,y)\psi_i(y)\,\mathrm{d}\mu_y\right)\varphi_j(x)\,\mathrm{d}\mu_x\right]^2\\ &\le c_{\Lambda_2}\int_{X_s}\sum_{i\in t}\left(\int_{Y_t} f(x,y)\psi_i(y)\,\mathrm{d}\mu_y\right)^2\mathrm{d}\mu_x\\ &\le c_{\Lambda_1}c_{\Lambda_2}\int_{X_s}\int_{Y_t}|f(x,y)|^2\,\mathrm{d}\mu_y\,\mathrm{d}\mu_x = c_{\Lambda_1}c_{\Lambda_2}\|f\|^2_{L^2(X_s\times Y_t)}\end{aligned}$$

it follows that $\|\Lambda\|_{F\leftarrow L^2}\le c_1c_2$. We deduce that

$$\|A_b-\tilde{A}_b\|_F^2 = \|\Lambda(\kappa-\tilde{\kappa})\|_F^2 \le \|\Lambda\|^2_{F\leftarrow L^2}\|\kappa-\tilde{\kappa}\|^2_{L^2(X_s\times Y_t)} \le (\varepsilon c_1c_2)^2\mu(X_s)\mu(Y_t).$$

For collocation matrices we find

$$\begin{aligned}\|A_b-\tilde{A}_b\|_F^2 &= \sum_{i\in t}\|\Lambda_{2,s}[\kappa(\cdot,y_i)-\tilde{\kappa}(\cdot,y_i)]\|_2^2\\ &\le c_{\Lambda_2}^2\sum_{i\in t}\|\kappa(\cdot,y_i)-\tilde{\kappa}(\cdot,y_i)\|^2_{L^2(X_s)} \le c_{\Lambda_2}^2|t|\varepsilon^2\mu(X_s).\end{aligned}$$

□

The previous theorem provides error estimates for the entries and the Frobenius norm on each block of a partition. Estimates for the spectral norm can be obtained from $\|A_b\|_2\le\|A_b\|_F$ for all $b\in P$. According to Remark 1.6, a kernel function which can be approximated by an exponentially convergent degenerate kernel on $X_s\times Y_t$ therefore leads to an exponential decay of the singular values of the block A_{ts}.

Usually, global error estimates are more relevant than blockwise estimates. The blockwise error with respect to the Frobenius norm is carried over to the global estimate, while for the error in the spectral norm we have the following theorem. Note that the application of Theorem 2.16 leads to worse estimates.

Theorem 3.27. *Assume that*

$$|\kappa(x,y)-\tilde{\kappa}(x,y)|\le\varepsilon \quad \textit{for all } x\in X_s,\ y\in Y_t.$$

Then for Galerkin matrices it holds that

$$\|A-\tilde{A}\|_2\le\varepsilon c_{\Lambda_1}c_{\Lambda_2}\nu\mu(\Omega)$$

and in the case of collocation matrices one has

$$\|A-\tilde{A}\|_2\le\varepsilon c_{\Lambda_2}[\nu|I|\mu(\Omega)]^{1/2},$$

where ν is defined in (1.20).

Proof. According to the previous theorem we have for all $u\in\mathbb{R}^J$ and all $v\in\mathbb{R}^I$

$$
\begin{aligned}
|(v,(A-\tilde{A})u)| &= |\sum_{t\times s\in P}(v_t,(A_{ts}-\tilde{A}_{ts})u_s)| \le \sum_{t\times s\in P}|(v_t,(A_{ts}-\tilde{A}_{ts})u_s)| \\
&\le \sum_{t\times s\in P}\varepsilon c_{\Lambda_1}c_{\Lambda_2}[\mu(X_s)\mu(Y_t)]^{1/2}\|v_t\|_2\|u_s\|_2 \\
&\le \varepsilon c_{\Lambda_1}c_{\Lambda_2}\left(\sum_{t\times s\in P}\mu(X_s)\mu(Y_t)\right)^{1/2}\left(\sum_{t\times s\in P}\|v_t\|_2^2\|u_s\|_2^2\right)^{1/2} \\
&\le \varepsilon c_{\Lambda_1}c_{\Lambda_2}\nu\mu(\Omega)\left(\sum_{t\times s\in P}\|v_t\|_2^2\|u_s\|_2^2\right)^{1/2}.
\end{aligned}
$$

The previous estimate follows from $\sum_{t\times s\in P}\|u_s\|_2^2\|v_t\|_2^2 = \|u\|_2^2\|v\|_2^2$ and

$$
\sum_{t\times s\in P}\mu(X_s)\mu(Y_t) \le \left(\sum_{j\in J}\mu(X_j)\right)\left(\sum_{i\in I}\mu(Y_i)\right) \le \nu^2\mu^2(\Omega),
$$

which is due to (1.20). The estimate for collocation matrices can be obtained analogously. □

3.3.4.1 Accuracy of the Solution

The aim of numerical methods is to obtain an (approximate) solution. Up to now, we have only considered the approximation error of the matrix resulting from kernel approximation. In this paragraph we will estimate the impact of matrix approximation errors on the discrete solution u_h of $Au_h = f_h$. For this purpose, we assume that the bilinear form $a(v,w) := (\mathscr{A}v,w)_{L^2(\Omega)}$ with $\mathscr{A}: V \to V'$ is continuous and coercive; i.e.,

$$
|a(v,w)| \le \alpha\|v\|_V\|w\|_V \quad \text{and} \quad a(v,v) \ge \beta\|v\|_V^2
$$

for all $v,w \in V \subset L^2(\Omega)$ with positive constants α, β.

Let $V_h := \operatorname{span}\{\varphi_1,\dots,\varphi_n\}$ with $\|\varphi_i\|_{L^2} = 1$, $i = 1,\dots,n$, approximate V in the sense that

$$
\inf_{v_h\in V_h}\|v-v_h\|_V \le h^r\|v\|_W \quad \text{for all } v \in W, \tag{3.54}
$$

where $W \subset V$ is a space of higher regularity than V. We assume that $\|\cdot\|_{L^2(\Omega)} \le \|\cdot\|_V \le \|\cdot\|_W$. Let $A \in \mathbb{R}^{n\times n}$ with the entries

$$
a_{ij} = (\mathscr{A}\varphi_j,\varphi_i)_{L^2}
$$

be a Galerkin discretization of $\mathscr{A}$. Each element $u_h \in V_h$ has the representation

$$
u_h = \sum_{i=1}^{n} u_i\varphi_i.
$$

Under appropriate assumptions on V_h there is a constant $c_{V_h} > 0$ such that

$$\|u\|_2 \le c_{V_h} \|u_h\|_{L^2(\Omega)},$$

for all $u \in \mathbb{R}^n$ with components u_i, $i = 1, \dots, n$. This kind of estimate holds, for instance, if the elements of V_h are piecewise linears and if the underlying geometry is quasi-uniform; cf. [126, Thm. 8.8.1]. Let $\tilde{A} \in \mathbb{R}^{n \times n}$ approximate A such that

$$\|A - \tilde{A}\|_2 \le \varepsilon \tag{3.55}$$

and define the perturbed bilinear form

$$\tilde{a}(v_h, w_h) := \sum_{i,j=1}^{n} \tilde{a}_{ij} w_i v_j.$$

Lemma 3.28. *The bilinear form $\tilde{a}$ is continuous and coercive on $V_h \times V_h$ with constants $\tilde{\alpha} := \alpha + c_{V_h}^2 \varepsilon$ and $\tilde{\beta} := \beta - c_{V_h}^2 \varepsilon$ if $\varepsilon < \beta / c_{V_h}^2$.*

Proof. From

$$|a(v_h, w_h) - \tilde{a}(v_h, w_h)| = |w^T (A - \tilde{A}) v| \le \|A - \tilde{A}\|_2 \|v\|_2 \|w\|_2 \le c_{V_h}^2 \varepsilon \|v_h\|_V \|w_h\|_V$$

it follows that

$$|\tilde{a}(v_h, w_h)| \le |a(v_h, w_h)| + |a(v_h, w_h) - \tilde{a}(v_h, w_h)| \le (\alpha + c_{V_h}^2 \varepsilon) \|v_h\|_V \|w_h\|_V$$

and $|\tilde{a}(v_h, v_h)| \ge |a(v_h, v_h)| - |a(v_h, v_h) - \tilde{a}(v_h, v_h)| \ge (\beta - c_{V_h}^2 \varepsilon) \|v_h\|_V^2$. □

The previous lemma states that the approximation has to be accurate enough in order to guarantee coercivity, which for symmetric bilinear forms is equivalent to $\tilde{A}$ being positive definite. We remark that the methods from Sect. 2.5.1 can be used to preserve positivity also for arbitrary bad approximations.

The following theorem describes the influence of matrix approximation errors on the accuracy of the discrete solution.

Theorem 3.29. *Let $u \in W$ be the solution of $a(u, v) = l(v)$ for all $v \in V$ and let $\tilde{u}_h \in V_h$ be the solution of $\tilde{a}(\tilde{u}_h, v_h) = l(v_h)$ for all $v_h \in V_h$. Then*

$$\|u - \tilde{u}_h\|_V \le \tilde{\beta}^{-1} \left\{ \left[\tilde{\alpha} + \tilde{\beta} + c_{V_h}^2 \varepsilon \right] h^r + c_{V_h}^2 \varepsilon \right\} \|u\|_W.$$

Proof. The assertion follows from Lemma 3.28 and the first Strang lemma (cf. [69])

$$\begin{aligned}
\|u - \tilde{u}_h\|_V &\le \inf_{v_h \in V_h} \left\{ \left[1 + \frac{\tilde{\alpha}}{\tilde{\beta}} \right] \|u - v_h\|_V + \tilde{\beta}^{-1} \sup_{w_h \in V_h} \frac{|a(v_h, w_h) - \tilde{a}(v_h, w_h)|}{\|v_h\|_V \|w_h\|_V} \|v_h\|_V \right\} \\
&\le \tilde{\beta}^{-1} \left[\tilde{\beta} + \tilde{\alpha} + c_{V_h}^2 \varepsilon \right] \inf_{v_h \in V_h} \|u - v_h\|_V + \tilde{\beta}^{-1} c_{V_h}^2 \varepsilon \|u\|_V \\
&\le \tilde{\beta}^{-1} \left[\tilde{\beta} + \tilde{\alpha} + c_{V_h}^2 \varepsilon \right] h^r \|u\|_W + \tilde{\beta}^{-1} c_{V_h}^2 \varepsilon \|u\|_W,
\end{aligned}$$

where we have used (3.54). □

The previous theorem shows that the additional approximation error which results from matrix approximation cannot be noticed as long as ε is of the order of the finite element error h^r. It is therefore enough to prove matrix norm estimates (3.55) for instance with respect to the spectral norm or the Frobenius norm.

The construction of degenerate kernel approximants leads to low-rank matrices and the analytic error translates into an algebraic error by Theorem 3.27. Hence, the required rank for a prescribed matrix error can be calculated from a-priori estimates such as (3.29), (3.38), (3.45), and (3.53). Remark 3.9, however, shows that the actual rank might be much smaller. In the following section we will therefore tackle the problem of constructing low-rank approximants from the matrix entries directly without approximating the kernel function.

3.4 Adaptive Cross Approximation (ACA)

We have seen that the construction of low-rank approximants based on degenerate kernel approximation might lead to non-optimal results. Although this problem can be overcome by recompressing the approximant with the procedure from Sect. 2.6, there are several other disadvantages. The construction of degenerate kernels for the Lamé kernel is already cumbersome, but it might be even more complicated if, for instance, problems from anisotropic elasticity are to be solved efficiently. Even if it is possible to find a degenerate kernel approximant, its quality remains in question. With the **adaptive cross approximation** (ACA) algorithm from this section we present a completely different approach; see [17, 32, 26]. In contrast to other methods like fast multipole, panel clustering, etc., the low-rank approximant is not generated by approximating the kernel function of the integral operator. Instead, we will directly find a low-rank approximant from few of the original matrix entries of admissible blocks. Note that it is not necessary to build the whole matrix beforehand. The respective matrix entries can be computed on demand. Working on the matrix entries has the advantage that the rank of the approximation can be chosen adaptively while kernel approximation requires an a-priori choice, which is usually too large. The property that only original matrix entries are used is of great practical importance, because usually much effort has gone into efficient and reliable computer codes for their computation. The usage of kernel approximants, in contrast, requires a complete recoding.

The singular value decomposition would find the lowest rank that is required for a given accuracy. However, its computational complexity makes it unattractive for large-scale computations. ACA can be regarded as an efficient replacement which is tailored to kernels that are asymptotically smooth with respect to at least one variable. Note that the kernel function κ itself is not required, only the information that κ is in this class of functions is important. Numerical experiments show that ACA can often be successfully applied although the kernel does not satisfy this assumption.

The construction of efficient algebraic methods for the treatment of discrete integral operators is an active field of research. To name a few methods known from electromagnetic simulation, these include the multilevel matrix decomposition algorithm (MLMDA) [186], IES^3 [152, 108], and the single-level IE-*QR* algorithm [198]. MLMDA was found to be efficient only for planar objects in $\mathbb{R}^3$; cf. [209]. The rank-revealing process of IES^3 is done through a so-called "statistically determined rank-map", which is constructed using the kernel function. While IE-*QR* is a rank-revealing *QR* decomposition, it will be seen that ACA is a rank-revealing *LU* decomposition which compared with IE-*QR* does not require the computationally expensive Gram-Schmidt orthogonalization process; see [263] for a comparison of the latter two methods. For the application of ACA to various problems see [168, 169, 248, 172, 52, 140, 197, 34, 246, 245].

3.4.1 The Algorithm

We assume that a partition has been generated as in Chap. 1. Blocks $b \in P$ which do not satisfy (3.35) are generated and stored without approximation. Therefore, this case is not treated here. All other blocks $b \in P$ satisfy (3.35) and can be treated independently from each other. Therefore, in the rest of this section we focus on a single block $A \in \mathbb{R}^{m\times n}$.

The idea of the algorithm is as follows. Starting from $R_0 := A$, find a nonzero pivot in R_k, say (i_k, j_k), and subtract a scaled outer product of the i_kth row and the j_kth column:

$$R_{k+1} := R_k - [(R_k)_{i_k j_k}]^{-1} (R_k)_{1:m,j_k} (R_k)_{i_k,1:n}, \tag{3.56}$$

where we use the notations $(R_k)_{i,1:n}$ and $(R_k)_{1:m,j}$ for the ith row and the jth column of R_k, respectively. It will turn out that j_k should be chosen the maximum element in modulus of the i_kth row; i.e.,

$$|(R_{k-1})_{i_k j_k}| = \max_{j=1,\dots,n} |(R_{k-1})_{i_k j}|. \tag{3.57}$$

The choice of i_k will be treated in Sect. 3.4.3. Notice that the above recursion (3.56) is the algebraic version of the construction (3.47) of the sequence r_k.

Example 3.30. We apply two steps of equation (3.56) to the following matrix R_0. The bold entries are the chosen pivots.

$$R_0 = \begin{bmatrix} 0.431 & 0.354 & 0.582 & 0.417 & 0.455 \\ 0.491 & 0.396 & 0.674 & 0.449 & 0.427 \\ 0.446 & 0.358 & 0.583 & 0.413 & 0.441 \\ 0.380 & 0.328 & 0.557 & 0.372 & 0.349 \\ 0.412 & 0.340 & 0.516 & 0.375 & 0.370 \end{bmatrix} \overset{\substack{i_1=1 \\ j_1=3}}{\longrightarrow} \frac{1}{0.582} \begin{bmatrix} 0.582 \\ 0.674 \\ 0.583 \\ 0.557 \\ 0.516 \end{bmatrix} \begin{bmatrix} 0.431 \\ 0.354 \\ 0.582 \\ 0.417 \\ 0.455 \end{bmatrix}^T,$$

$$R_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -0.008 & -0.014 & 0 & -0.033 & -0.100 \\ 0.014 & 0.003 & 0 & -0.004 & -0.014 \\ -0.032 & -0.011 & 0 & -0.026 & -0.087 \\ 0.029 & 0.025 & 0 & 0.005 & -0.034 \end{bmatrix} \overset{\substack{i_2=2 \\ j_2=5}}{\longrightarrow} \frac{1}{-0.1} \begin{bmatrix} 0 \\ -0.100 \\ -0.014 \\ -0.087 \\ -0.034 \end{bmatrix} \begin{bmatrix} -0.008 \\ -0.014 \\ 0 \\ -0.033 \\ -0.100 \end{bmatrix}^T,$$

$$R_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0.016 & 0.005 & 0 & 0.000 & 0 \\ -0.02 & 0.001 & 0 & 0.002 & 0 \\ 0.032 & 0.030 & 0 & 0.017 & 0 \end{bmatrix} \overset{\substack{i_3=3 \\ j_3=1}}{\longrightarrow} \frac{1}{0.016} \begin{bmatrix} 0 \\ 0 \\ 0.016 \\ -0.02 \\ 0.032 \end{bmatrix} \begin{bmatrix} 0.016 \\ 0.005 \\ 0 \\ 0.000 \\ 0 \end{bmatrix}^T$$

Apparently, the size of the entries decreases from step to step.

Since in the kth step only the entries in the j_kth column and the i_kth row of R_k are used to compute R_{k+1}, there is no need to build the whole matrix R_k. In particular this means that only few of the original entries of A have to be computed. Taking advantage of this, the following algorithm is an efficient reformulation of (3.56). Note that the vectors u_k and $\tilde{v}_k$ coincide with $(R_{k-1})_{1:m,j_k}$ and $(R_{k-1})^T_{i_k,1:n}$, respectively.

Let $k = 1$; $Z = \varnothing$;
repeat
  find i_k as described in Sect. 3.4.3
  $\tilde{v}_k := a_{i_k,1:n}$
  for $\ell = 1,\ldots,k-1$ **do** $\tilde{v}_k := \tilde{v}_k - (u_\ell)_{i_k} v_\ell$
  $Z := Z \cup \{i_k\}$
  if $\tilde{v}_k$ does not vanish **then**
    $j_k := \operatorname{argmax}_{j=1,\ldots,n} |(\tilde{v}_k)_j|$; $v_k := (\tilde{v}_k)^{-1}_{j_k} \tilde{v}_k$
    $u_k := a_{1:m,j_k}$
    for $\ell = 1,\ldots,k-1$ **do** $u_k := u_k - (v_\ell)_{j_k} u_\ell$.
    $k := k+1$
  endif
until the stopping criterion (3.58) is fulfilled or $Z = \{1,\ldots,m\}$

Algorithm 3.1: Adaptive Cross Approximation (ACA).

The vanishing rows of the R_k's are collected in the set Z. If the i_kth row of R_k is nonzero and hence is used as v_k, it is also added to Z since the i_kth row of R_{k+1} will vanish. The matrix $S_k := \sum_{\ell=1}^k u_\ell v_\ell^T$ will be used as an approximation of $A = S_k + R_k$. Obviously, the rank of S_k is bounded by k. Although ACA already provides a good approximant (see Fig. 3.2) it may be worth reducing the required rank by the recompression procedure from Sect. 2.6. The method proposed in Sect. 3.5 recompresses the approximation generated by ACA such that the overall complexity of $\mathcal{H}^2$-matrices can be achieved. Note that it is obvious how Algorithm 3.1 has to be modified if it is applied to complex matrices.

Let $\varepsilon > 0$ be given. The following condition on k

$$\|u_{k+1}\|_2 \, \|v_{k+1}\|_2 \le \frac{\varepsilon(1-\eta)}{1+\varepsilon} \, \|S_k\|_F \tag{3.58}$$

can be used as a stopping criterion. Assume that $\|R_{k+1}\|_F \leq \eta \|R_k\|_F$ with η from (3.35), then

$$\|R_k\|_F \leq \|R_{k+1}\|_F + \|u_{k+1} v_{k+1}^T\|_F \leq \eta \|R_k\|_F + \|u_{k+1}\|_2 \|v_{k+1}\|_2.$$

Hence,

$$\|R_k\|_F \leq \frac{1}{1-\eta} \|u_{k+1}\|_2 \|v_{k+1}\|_2 \leq \frac{\varepsilon}{1+\varepsilon} \|S_k\|_F \leq \frac{\varepsilon}{1+\varepsilon} (\|A\|_F + \|R_k\|_F).$$

From the previous estimate we obtain $\|R_k\|_F \leq \varepsilon \|A\|_F$; i.e., condition (3.58) guarantees a relative approximation error ε. Hence, the rank required to guarantee a prescribed accuracy can be found adaptively.

Due to (1.4), the Frobenius norm of S_k can be computed with $\mathcal{O}(k^2(m+n))$ complexity. Therefore, the amount of numerical work required by Algorithm 3.1 is of the order $|Z|^2(m+n)$.

Remark 3.31. If the costs for generating the matrix entries dominate the algebraic transformations of Algorithm 3.1, then its complexity scales like $|Z|(m+n)$.

3.4.2 Error Analysis

In order to estimate the efficiency of ACA, we have to find a bound for the norm of the remainder R_k. The analysis in the case of the Nyström method was done in [17], collocation matrices were treated in [18, 32]. The proofs rely on the fact that collocation and Nyström matrices arise from evaluating functions at given points. Therefore, interpolation results were applied. For Galerkin matrices an analogous result was doubtful due to the variational character of the formulation. In the following section we will present a short unified proof which also covers the case of Galerkin matrices; see [26]. For the convergence analysis we will assume that the kernel function κ is asymptotically smooth with respect to y, which can be guaranteed for elliptic problems due to Remark 3.1 and Lemma 3.5.

Without loss of generality we assume in the rest of this section that for the pivotal indices i_ℓ and j_ℓ it holds that $i_\ell = j_\ell = \ell$, $\ell = 1, \ldots, k$. Then A has the decomposition

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad A_{11} \in \mathbb{R}^{k \times k}, \tag{3.59}$$

where only the matrix blocks A_{11}, A_{12}, and A_{21} have been used in Algorithm 3.1. Note that the (large) block $A_{22} \in \mathbb{R}^{(m-k)\times(n-k)}$ has never been touched. Since the determinant of A_{11} is the product of the pivots, A_{11} is invertible and we can express the remainder R_k of the approximation in terms of the original matrix A.

Lemma 3.32. *For R_k it holds that*

$$R_k = A - \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix} A_{11}^{-1} \begin{bmatrix} A_{11} & A_{12} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & C_k \end{bmatrix},$$

where $C_k := A_{22} - A_{21}A_{11}^{-1}A_{12}$ *is the Schur complement of* A_{11} *in* A.

Proof. The assertion is obvious for $k = 1$. Assume that it holds for k and let A be decomposed in the following way

$$A = \begin{bmatrix} A_{11} & w & B \\ v^T & \alpha & y^T \\ C & x & D \end{bmatrix}, \quad A_{11} \in \mathbb{R}^{k\times k},$$

with vectors $x \in \mathbb{R}^{m-k-1}$, $y \in \mathbb{R}^{n-k-1}$, $v, w \in \mathbb{R}^k$, and $\alpha \in \mathbb{R}$. From (3.56) we see that

$$R_{k+1} = A - \begin{bmatrix} A_{11} & w \\ v^T & \alpha \\ C & x \end{bmatrix} \begin{bmatrix} A_{11}^{-1} + \gamma A_{11}^{-1} w v^T A_{11}^{-1} & -\gamma A_{11}^{-1} w \\ -\gamma v^T A_{11}^{-1} & \gamma \end{bmatrix} \begin{bmatrix} A_{11} & w & B \\ v^T & \alpha & y^T \end{bmatrix},$$

where $\gamma = (\alpha - v^T A_{11}^{-1} w)^{-1}$ and $\alpha - v^T A_{11}^{-1} w$ is the pivot, which is chosen nonzero. Since

$$\begin{bmatrix} A_{11} & w \\ v^T & \alpha \end{bmatrix}^{-1} = \begin{bmatrix} A_{11}^{-1} + \gamma A_{11}^{-1} w v^T A_{11}^{-1} & -\gamma A_{11}^{-1} w \\ -\gamma v^T A_{11}^{-1} & \gamma \end{bmatrix},$$

we obtain the desired result. The second part of the assertion is obvious. □

From Algorithm 3.1 it can be seen that the number $k' := |Z|$ of zero columns in R_k may be larger than k; i.e., $k' \geq k$. This happens if in the ℓth step the algorithm comes across a zero vector $\tilde{v}_\ell$ and has to continue with another row. Without loss of generality we assume that the indices corresponding to a zero row met in Algorithm 3.1 are $\{k+1, \ldots, k'\}$. Let A_{21} and A_{22} from (3.59) be decomposed in the following way

$$A_{21} = \begin{bmatrix} \hat{A}_{21} \\ \check{A}_{21} \end{bmatrix}, \quad A_{22} = \begin{bmatrix} \hat{A}_{22} \\ \check{A}_{22} \end{bmatrix}, \quad \hat{A}_{21} \in \mathbb{R}^{(k'-k)\times k}, \hat{A}_{22} \in \mathbb{R}^{(k'-k)\times(n-k)}.$$

It is remarkable that although the approximant S_k has rank at most k, not k but k' will determine the accuracy of the approximation as can be seen from the following lemma, which will be used to estimate the norm of the Schur complement and hence the norm of the remainder R_k. This observation is also of practical importance. Since a zero row is not lost for the approximation accuracy, the problem of finding a nonzero pivot will not lead to the computation of the whole matrix.

Lemma 3.33. *Let* $X \in \mathbb{R}^{(m-k)\times k'}$ *be arbitrary, then*

$$C_k = \left\{ A_{22} - X \begin{bmatrix} A_{12} \\ \hat{A}_{22} \end{bmatrix} \right\} - \left\{ A_{21} - X \begin{bmatrix} A_{11} \\ \hat{A}_{21} \end{bmatrix} \right\} A_{11}^{-1} A_{12}. \tag{3.60}$$

Proof. It is easy to check that a zero row in R_ℓ will remain zero in R_k, $k \geq \ell$. Hence, $(R_k)_{i,1:n} = 0$, $i = k+1, \ldots, k'$. From Lemma 3.32 it follows that

$$\hat{A}_{22} = \hat{A}_{21} A_{11}^{-1} A_{12}.$$

Adding and subtracting

$$X \begin{bmatrix} A_{12} \\ \hat{A}_{22} \end{bmatrix} = X \begin{bmatrix} A_{11} \\ \hat{A}_{21} \end{bmatrix} A_{11}^{-1} A_{12}$$

to and from C_k we end up with what we were after. □

The expressions from (3.60) appearing in curly braces will be estimated by relating them to interpolation errors. What remains for an estimate of C_k, is a bound on the size of the coefficients $(A_{11}^{-1}A_{12})_{ij}$. By Cramer's rule it holds that

$$(A_{11}^{-1}A_{12})_{ij} = \frac{\det(a_1, \ldots, a_{i-1}, a'_j, a_{i+1}, \ldots, a_k)}{\det A_{11}},$$

where a_ℓ, $\ell = 1, \ldots, k$, are the columns of A_{11} and a'_j is the jth column of A_{12}. The coefficients $(A_{11}^{-1}A_{12})_{ij}$ in (3.60) should be as small as possible for a small norm of C_k. The optimal choice of the pivots would be a submatrix A_{11} having maximum determinant in modulus. The **method of pseudo-skeletons** (cf. [252, 111, 110]) is based on this pivoting strategy. To find such a submatrix in A with reasonable effort, however, seems to be impossible. The pivoting strategy used in Algorithm 3.1 gives the following bound on the size of the entries in $A_{11}^{-1}A_{12}$. The proof can be done analogously to the proof of Lemma 3.25.

Lemma 3.34. *Assume that in each step j_k is chosen so that* (3.57) *is satisfied. Then for $i = 1, \ldots, k$ and $j = 1, \ldots, n-k$ it holds that*

$$|\det(a_1, \ldots, a_{i-1}, a'_j, a_{i+1}, \ldots, a_k)| \leq 2^{k-i} |\det A_{11}|.$$

We are now ready to estimate the remainder R_k. For this purpose, the entries of R_k will be estimated by the approximation error

$$F_{ts}^{\Xi} := \max_{j \in s} \inf_{p \in \operatorname{span} \Xi} \|\mathscr{A} \Lambda_{2,j}^* - p\|_{\infty, Y_t} \tag{3.61}$$

in an arbitrary system of functions $\Xi := \{\xi_1, \ldots, \xi_{k'}\}$ with $\xi_1 = 1$. Note that $\mathscr{A}\Lambda_{2,j}^*$ is an asymptotically smooth function due to $\operatorname{supp} \Lambda_{2,j}^* = X_j$. In Sect. 3.3 we have presented systems Ξ together with their approximation errors. In the next theorem the approximation error associated with Algorithm 3.1 applied to collocation and Nyström matrices, i.e., the case $\Lambda_{1,i} f = f(y_i)$, is estimated. In Theorem 3.37 Galerkin matrices will be considered.

For the following theorem we assume as in Sect. 3.3.3 the unisolvency of the system Ξ in the nodes y_i, $i = 1, \ldots, k'$; i.e., the Vandermonde determinant does not vanish

$$\det [\xi_j(y_i)]_{i,j=1,\ldots,k'} \neq 0. \tag{3.62}$$

This condition will be satisfied by the choice of rows i_k in Sect. 3.4.3.

Theorem 3.35. *Let $\Lambda_{1,i} f = f(y_i)$, $i = 1,\dots,m$. Then for $i = 1,\dots,m$ and $j = 1,\dots,n$ it holds that*

$$|(R_k)_{ij}| \le 2^k (1 + \|\mathfrak{I}_{k'}^{\Xi}\|) F_{ts}^{\Xi}, \tag{3.63}$$

where F_{ts}^{Ξ} is defined in (3.61).

Proof. For the entries of the $(m-k)\times(n-k)$ matrix $A_{22} - X \begin{bmatrix} A_{12} \\ \hat{A}_{22} \end{bmatrix}$ it holds that

$$(A_{22} - X \begin{bmatrix} A_{12} \\ \hat{A}_{22} \end{bmatrix})_{ij} = \mathscr{A}\Lambda_{2,j}^*(y_i) - \sum_{\ell=1}^{k'} L_\ell^{\Xi}(y_i)\mathscr{A}\Lambda_{2,j}^*(y_\ell),$$

where we have chosen $X_{i\ell} = L_\ell^{\Xi}(y_i)$. Similar to (3.50) we have

$$|\mathscr{A}\Lambda_{2,j}^*(y_i) - \sum_{\ell=1}^{k'} L_\ell^{\Xi}(y_i)\mathscr{A}\Lambda_{2,j}^*(y_\ell)| \le (1 + \|\mathfrak{I}_{k'}^{\Xi}\|) \inf_{p\in \operatorname{span}\Xi} \|\mathscr{A}\Lambda_{2,j}^* - p\|_{\infty,Y_t}.$$

The same kind of estimate holds for the entries of $A_{21} - X \begin{bmatrix} A_{11} \\ \hat{A}_{21} \end{bmatrix}$. Therefore, from Lemma 3.34 we obtain

$$|(R_k)_{ij}| \le 2^k (1 + \|\mathfrak{I}_{k'}^{\Xi}\|) \max_{j\in s} \inf_{p\in \operatorname{span}\Xi} \|\mathscr{A}\Lambda_{2,j}^* - p\|_{\infty,Y_t}.$$

□

Galerkin matrices need a different treatment. For the localizers $\Lambda_{1,i}$, $i = 1,\dots,k'$, corresponding to the first k' rows in A we assume the following generalization of (3.62)

$$\det[\Lambda_{1,i}\xi_j]_{i,j=1,\dots,k'} \ne 0, \tag{3.64}$$

which will be guaranteed by the choice of pivoting rows i_k in Sect. 3.4.3.

The localizers $\Lambda_{1,i}$ in the case of Galerkin matrices read

$$\Lambda_{1,i} f = \int_\Omega f(y)\psi_i(y)\,\mathrm{d}\mu_y, \quad i = 1,\dots,m. \tag{3.65}$$

In the following lemma we will construct functions $\sum_{\ell=1}^{k'} c_\ell^{(i)} \psi_\ell / \|\psi_\ell\|_{L^1}$ for each ψ_i such that some kind of vanishing moments property

$$\int_\Omega \left(\frac{\psi_i}{\|\psi_i\|_{L^1}} - \sum_{\ell=1}^{k'} c_\ell^{(i)} \frac{\psi_\ell}{\|\psi_\ell\|_{L^1}} \right) q = 0 \quad \text{for all } q \in \operatorname{span}\Xi \tag{3.66}$$

holds.

Lemma 3.36. *Assume that condition* (3.64) *holds. Then for each $i \in \{1,\dots,m\}$ there are uniquely determined coefficients $c_\ell^{(i)}$, $\ell = 1,\dots,k'$, such that* (3.66) *holds.*

Proof. For each $q \in \operatorname{span}\Xi$ there are coefficients α_j, $j = 1,\dots,k'$, such that

$$q = \sum_{j=1}^{k'} \alpha_j \xi_j. \tag{3.67}$$

Since the matrix $(\Lambda_{1,i}\xi_j)_{ij}$ is non-singular, the linear system

$$\sum_{\ell=1}^{k'} c_\ell^{(i)} \frac{\Lambda_{1,\ell}\xi_j}{\|\psi_\ell\|_{L^1}} = \frac{\Lambda_{1,i}\xi_j}{\|\psi_i\|_{L^1}}, \quad j = 1,\dots,k',$$

is uniquely solvable with respect to $c_\ell^{(i)}$, $\ell = 1,\dots,k'$, for each $i \in \{1,\dots,m\}$. The assertion follows from (3.67) and the linearity of the operators $\Lambda_{1,i}$. □

The coefficients $c_\ell^{(i)}$ from the previous lemma depend on the shape of the grid, but they do depend neither on the kernel function κ nor on the size of the finite elements. The following theorem treats the case of Galerkin matrices.

Theorem 3.37. *Let $\Lambda_{1,i}$ be defined as in* (3.65). *Then for $i = 1,\dots,m$ and $j = 1,\dots,n$ it holds that*

$$|(R_k)_{ij}| \le 2^k (1 + \|\mathfrak{I}_{k'}^{\Xi}\|) \left(1 + \sum_{\ell=1}^{k'} |c_\ell^{(i)}|\right) \|\psi_i\|_{L^1} F_{ts}^{\Xi}. \tag{3.68}$$

Proof. Since $\mathfrak{I}_{k'}^{\Xi} \mathscr{A} \Lambda_{2,j}^* \in \operatorname{span}\Xi$, according to Lemma 3.36 there are coefficients $c_\ell^{(i)}$ such that

$$\int_\Omega \mathfrak{I}_{k'}^{\Xi} \mathscr{A} \Lambda_{2,j}^*(y) \frac{\psi_i(y)}{\|\psi_i\|_{L^1}} \,\mathrm{d}\mu_y = \sum_{\ell=1}^{k'} c_\ell^{(i)} \int_\Omega \mathfrak{I}_{k'}^{\Xi} \mathscr{A} \Lambda_{2,j}^*(y) \frac{\psi_\ell(y)}{\|\psi_\ell\|_{L^1}} \,\mathrm{d}\mu_y.$$

Let $X \in \mathbb{R}^{(m-k)\times k'}$ be the matrix with entries

$$X_{i\ell} := \frac{\|\psi_i\|_{L^1}}{\|\psi_\ell\|_{L^1}} c_\ell^{(i)}, \quad i = 1,\dots,m-k,\ \ell = 1,\dots,k'.$$

Then for the entries of the $(m-k)\times(n-k)$ matrix $A_{22} - X \begin{bmatrix} A_{12} \\ \hat{A}_{22} \end{bmatrix}$ it holds that

$$
\begin{aligned}
&(A_{22} - X \begin{bmatrix} A_{12} \\ \hat{A}_{22} \end{bmatrix})_{ij} \\
&\quad = \int_\Omega \mathscr{A}\Lambda_{2,j}^*(y)\,\psi_i(y)\,\mathrm{d}\mu_y - \sum_{\ell=1}^{k'} c_\ell^{(i)} \frac{\|\psi_i\|_{L^1}}{\|\psi_\ell\|_{L^1}} \int_\Omega \mathscr{A}\Lambda_{2,j}^*(y)\,\psi_\ell(y)\,\mathrm{d}\mu_y \\
&\quad = \int_\Omega (\mathscr{I} - \mathfrak{I}_{k'}^{\Xi})\mathscr{A}\Lambda_{2,j}^*(y)\,\psi_i(y)\,\mathrm{d}\mu_y \\
&\qquad - \sum_{\ell=1}^{k'} c_\ell^{(i)} \frac{\|\psi_i\|_{L^1}}{\|\psi_\ell\|_{L^1}} \int_\Omega (\mathscr{I} - \mathfrak{I}_{k'}^{\Xi})\mathscr{A}\Lambda_{2,j}^*(y)\,\psi_\ell(y)\,\mathrm{d}\mu_y \\
&\quad = \int_\Omega E_{k'}^{\Xi}[\mathscr{A}\Lambda_{2,j}^*](y)\,\psi_i(y)\,\mathrm{d}\mu_y - \sum_{\ell=1}^{k'} c_\ell^{(i)} \frac{\|\psi_i\|_{L^1}}{\|\psi_\ell\|_{L^1}} \int_\Omega E_{k'}^{\Xi}[\mathscr{A}\Lambda_{2,j}^*](y)\,\psi_\ell(y)\,\mathrm{d}\mu_y,
\end{aligned}
$$

where $E_{k'}^{\Xi}[f] := f - \mathfrak{I}_{k'}^{\Xi} f$. From (3.50) we have

$$
\int_\Gamma |E_{k'}^{\Xi}[\mathscr{A}\Lambda_{2,j}^*](y)|\,|\psi_i(y)|\,\mathrm{d}\mu_y \le (1 + \|\mathfrak{I}_{k'}^{\Xi}\|)\|\psi_i\|_{L^1} \inf_{p\in\Xi} \|\mathscr{A}\Lambda_{2,j}^* - p\|_{\infty,Y_t}.
$$

The same kind of estimate holds for $A_{21} - X \begin{bmatrix} A_{11} \\ \hat{A}_{21} \end{bmatrix}$. From Lemma 3.34 we obtain

$$
|(R_k)_{ij}| \le 2^k (1 + \|\mathfrak{I}_{k'}^{\Xi}\|) \left(1 + \sum_{\ell=1}^{k'} |c_\ell^{(i)}|\right) \|\psi_i\|_{L^1} \max_{j\in s} \inf_{p\in\Xi} \|\mathscr{A}\Lambda_{2,j}^* - p\|_{\infty,Y_t}.
$$

□

The similarity of Algorithm 3.1 and the *LU* factorization can be seen from the following representation

$$
R_k = (I - \gamma_k R_{k-1} e_k e_k^T) R_{k-1} = L_k R_{k-1}
$$

with the $m \times m$ matrix L_k defined by

$$
L_k = \begin{bmatrix}
1 & & & & & \\
& \ddots & & & & \\
& & 1 & & & \\
& & & 0 & & \\
& & & -\frac{(R_{k-1})_{k+1,k}}{(R_{k-1})_{kk}} & 1 & \\
& & & \vdots & & \ddots \\
& & & -\frac{(R_{k-1})_{mk}}{(R_{k-1})_{kk}} & & 1
\end{bmatrix},
$$

which differs from a Gauß matrix only in the position (k,k). It is known that during the *LU* decomposition the so-called *growth of entries* may happen; cf. [106]. Note

that this is reflected by the factor 2^k in (3.63) and (3.68). However, this growth is also known to be rarely observable in practice.

3.4.3 The Right Choice of Rows

We have seen that the choice of columns j_k from condition (3.57) is important for the boundedness of the coefficient $A_{11}^{-1}A_{12}$ in the error estimate. The choice of the rows i_k will guarantee condition (3.62) or more generally (3.64), which the interpolation in the system Ξ relies on. Satisfying this condition is indispensable as can be seen from the following example.

Example 3.38. The evaluation of the double-layer potential in $\mathbb{R}^3$

$$(\mathcal{K}\varphi_j)(y) = \frac{1}{4\pi}\int_\Gamma \frac{\nu_x \cdot (y-x)}{\|x-y\|^3}\varphi_j(x)\,\mathrm{d}s_x$$

at y_i, where φ_j are defined on the left cluster and y_i are located on the right cluster of Fig. 3.1, leads to a reducible matrix A having the structure

$$A = \begin{bmatrix} 0 & A_{12} \\ A_{21} & 0 \end{bmatrix}.$$

The zero block in the first block row of A is caused by the interaction of domain D_1 with D_3 and the other block by the interaction of domain D_2 with D_4 lying on a common plane, respectively. If ACA is applied to A with a starting pivot from the rows and columns of A_{12}, then the next nonzero pivot can only be found within A_{12}. Hence, the algorithm stays within A_{12} and the stopping criterion is not able to notice the untouched entries of A_{21}.

The reason for this is that the interpolation points y_{i_k} are exclusively chosen from the plane described by D_3 if ACA is applied to A_{12}. As a consequence, the Vandermonde determinant will vanish for this choice of interpolation points and the error estimate (3.63) does not hold for points $y_i \in D_4$. However, it remains valid for points $y_i \in D_3$ since the interpolation problem then reduces to a problem in $\mathbb{R}^2$ for which (3.62) can be guaranteed by an appropriate choice of $y_{i_k} \in D_3$.

The example above underlines the importance of the row choice, namely condition (3.64) in the convergence analysis of ACA. Assuming that ACA works without this condition may lead to a certain confusion; see [43].

We present two methods, one of which should be used to circumvent the mentioned difficulty. The first is an easy and obvious heuristic which seems to work reliably. The second method explicitly guarantees (3.64) such that ACA will converge as predicted by our analysis.

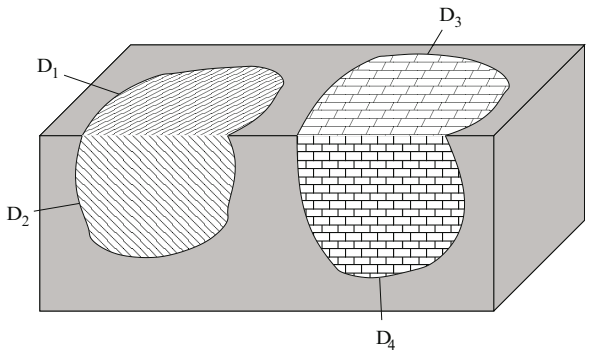


Fig. 3.1 Two clusters leading to a reducible matrix block.

A heuristic way of choosing the pivots

For each row and each column of A we introduce a counter which reflects the number of successful approximations applied to the respective row or column. An approximation is considered as successful for row i if $|(u_k)_i|$ is of the order of the estimated error $\|R_{k-1}\|_F$. The counter of the column j is increased if $|(v_k)_j|$ has the size of $\|R_{k-1}\|_F$. With these counters one can easily detect those rows and columns for which the error estimator is not reliable. Pivoting to the latter rows and columns approximates the unattended rows and columns.

A rigorous choice of pivots

Although the problem described in Example 3.38 can be overcome by the mentioned trick, for a rigorous analysis that works in a general setting we have to guarantee that (3.64) is satisfied by the choice of pivots. The first row i_1 can be arbitrarily chosen, because we have assumed that $\xi_1(x) = 1$. Assume that $i_1, \dots, i_k \in \{1, \dots, m\}$ and $j'_1, \dots, j'_k \in \{1, \dots, n\}$ have already been found such that (3.64) holds, i.e., that the Vandermonde matrix $W_k \in \mathbb{R}^{k \times k}$ having the entries

$$(W_k)_{\mu\nu} = \Lambda_{1,i_\mu} \xi_{j'_\nu}, \quad \mu, \nu = 1, \dots, k,$$

is non-singular. Note that the indices j'_k may differ from the indices j_k used in the previous section. The new pivot (i_{k+1}, j'_{k+1}) has to guarantee that W_{k+1} is non-singular, too.

Assume that the normalized LU decomposition of the $k \times k$ matrix $W_k = L_k U_k$ has been computed. Let j'_{k+1} be the next column which has not yet been investigated. Then

$$\det W_{k+1} = \left[c - (U_k^{-T} a)^T (L_k^{-1} b)\right] \det W_k,$$

where

$$W_{k+1} = \begin{bmatrix} W_k & b \\ a^T & c \end{bmatrix}$$

and $a = [\Lambda_{1,i_{k+1}} \xi_{j'_\ell}]_{\ell=1,\dots,k}$, $b = [\Lambda_{1,i_\ell} \xi_{j'_{k+1}}]_{\ell=1,\dots,k}$, and $c = \Lambda_{1,i_{k+1}} \xi_{j'_{k+1}}$. Hence, any index i_{k+1} satisfying $(U_k^{-T} a)^T (L_k^{-1} b) \neq c$ can be chosen as the new pivot. For stability, however, it is wise to choose i_{k+1} such that $|c - (U_k^{-T} a)^T (L_k^{-1} b)|$ is maximized. Testing a new row i_{k+1} requires k^2 operations. Since possibly all remaining $m-k$ rows have to be tested, finding i_{k+1} requires $k^2(m-k)$ operations. The LU decomposition of W_{k+1} is then given by

$$W_{k+1} = \begin{bmatrix} L_k & 0 \\ a^T U_k^{-1} & 1 \end{bmatrix} \begin{bmatrix} U_k & L_k^{-1} b \\ 0 & c - (U_k^{-T} a)^T (L_k^{-1} b) \end{bmatrix}.$$

It may happen that columns of $\Lambda_{1,i} \xi_j$ are linearly dependent. If $\Lambda_{1,i} f = f(y_i)$ with points y_i, $i = 1, \dots, m$, then this is equivalent to all points y_i lying on a hypersurface $H := \{x \in \mathbb{R}^d : P(x) = 0\}$, where $P \in \operatorname{span} \Xi$. In such a situation, for certain columns no $i_{k+1} \in \{1, \dots, m\}$ can be found such that $(U_k^{-T} a)^T (L_k^{-1} b) \neq c$. In this case let j'_{k+1} be the first column which allows to choose i_{k+1} such that $(U_k^{-T} a)^T (L_k^{-1} b) \neq c$. Note that the number of investigated columns is bounded by the number of columns required in the case of linear independent columns. This can be seen from

$$\operatorname{span} \Xi = \operatorname{span} \Xi' \quad \text{on } H,$$

where $\Xi' := \{\xi_{j'_1}, \dots, \xi_{j'_{k'}}\}$. The previous equation shows that k' columns give the same error $F_{ts}^{\Xi'}$ as all k. The same arguments hold if $\Lambda_{1,i} f = \int_\Omega f \psi_i \, d\mu$.

Example 3.39. Let $d = 2$ and $\Xi = \{1, x_1, x_2, x_1^2, x_1 x_2, x_2^2, \dots\}$. If all points y_i are located on the x-axis, then $j'_1 = 1$, $j'_2 = 2$, $j'_3 = 4$ and so on. Hence, this pivoting strategy will detect the reduced dimensionality of the problem and will implicitly use $\Xi' := \{1, x_1, x_1^2, \dots\}$.

As a result of the above procedure, we obtain two sequences i_ℓ and j'_ℓ, $\ell = 1, \dots, k'$, such that (3.64) is satisfied; i.e.,

$$\det [\Lambda_{1,i_\mu} \xi_{j'_\nu}]_{1 \leq \mu, \nu \leq k'} \neq 0.$$

Example 3.40. We return to the problem from Example 3.38. Assume that three pivots have been chosen from D_3. If i_4 is chosen from the same set, then one easily checks that

$$\det [\xi_j(y_{i_\ell})]_{1 \leq \ell, j \leq 4} = 0,$$

where $\xi_1(x) = 1$, $\xi_2(x) = x_1$, $\xi_3(x) = x_2$, and $\xi_4(x) = x_3$. There is however $y_{i_4} \in D_4$ which satisfies $\det [\xi_j(y_{i_\ell})]_{1 \leq \ell, j \leq 4} \neq 0$. Hence, ACA will automatically choose the forth pivot from the row indices of the block A_{22}.

Note that the above pivoting criterion requires only geometrical information and information about $\Lambda_{1,i}$, i.e, whether a collocation or a Galerkin method is employed. The kernel function, however, is still not required. The method from [43] is based

on explicit kernel approximation using the construction from Sect. 3.3.3. As a consequence, this method cannot use the original matrix entries.

The choice of the initial row

We have seen that the first row i_1 can be arbitrarily chosen. The efficiency of ACA can however be improved by the following choice. Due to the Definition 3.2 of asymptotic smoothness, each kernel function $\kappa(x,y)$ is almost linear with respect to y on $X_s \times Y_t$. Hence, it seems desirable to minimize the expression

$$\max_{y\in Y_t} |\kappa(x,y) - \kappa(x,z)| = \max_{y\in Y_t} \left| \int_y^z \partial_y \kappa(x,\xi)\, \mathrm{d}\xi \right| \leq c \max_{y\in Y_t} \|y-z\|$$

appearing in the remainder after the first step of ACA. The minimum of the upper bound $\max_{y\in Y_t} \|y-z\|$ is attained for the Chebyshev center $z = \xi_{Y_t}$ of Y_t. Since the Chebyshev center of a polygonal set is quite expensive to compute, we use the centroid m_t of Y_t instead. Hence, from these arguments it seems promising to choose i_1 so that the center z_{i_1} of Y_{i_1} is closest to m_t.

In Fig. 3.2 we compare this strategy with the "old" strategy in which i_1 is chosen so that z_{i_1} is closest to the centroid of X_s. The matrix which these methods are applied to arises from evaluating $\|x-y\|^{-1}$ at two sets of points having a distance which is large compared with their diameters. In addition, Fig. 3.2 shows the accuracy of approximations obtained from the multipole expansion (see Example 3.10) and from the singular value decomposition. The first k singular triplets give

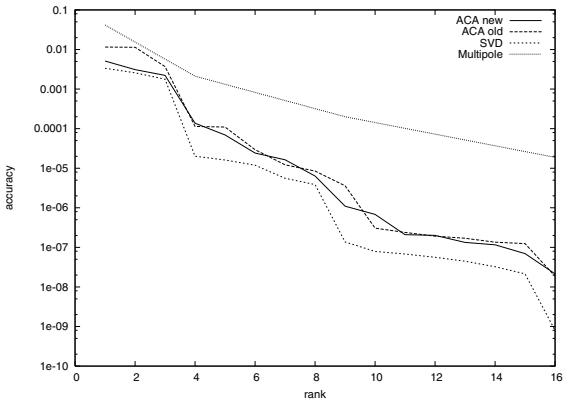


Fig. 3.2 Accuracy of different approximants.

provably (cf. Theorem 1.7) the best accuracy among all rank-k approximants. The influence of the new strategy on the quality of the approximation can be realized in the first four steps, which are often sufficient to obtain a reasonable accuracy. In

this part the new version gives almost optimal approximation. For all other steps the old and the new version behave almost the same. The quality of the approximant generated from the multipole expansion is significantly worse.

Alternative pivoting strategies

In [115] an alternative pivoting strategy for ACA is proposed which uses an initially chosen reference cross for choosing the pivots i_k and j_k. Notice this so-called ACA+ method is not able to handle the problem from Example 3.38 in general. In Fig. 3.3 we compare the quality of the approximation obtained by ACA+ and ACA. Since the convergence of ACA+ depends on the chosen reference cross and since there is no rule to choose it, we show the results for two randomly chosen reference crosses. The respective results are labeled "ACA+ 1" and "ACA+ 2". While the errors of "ACA+ 1" almost coincide with those of ACA, the convergence of "ACA+ 2" is not as "smooth". For instance, the rank-8 approximation leads to an error that is as large as the error of the rank-3 approximation. The previous observation is a hurdle to an efficient estimation of the error using (3.58).

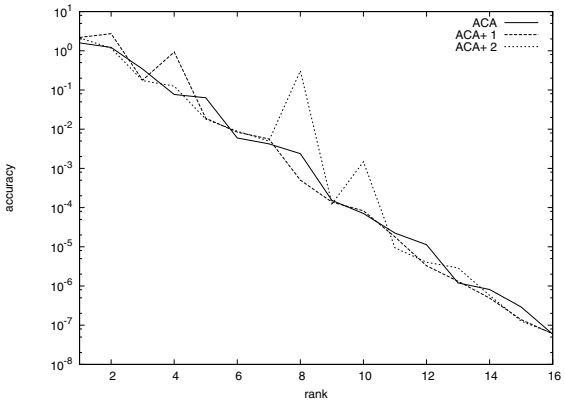


Fig. 3.3 Comparison of ACA and ACA+.

3.4.4 Overall Complexity

Finally, we will estimate the overall cost of generating the $\mathscr{H}$-matrix approximant to $A \in \mathbb{R}^{I \times J}$ with $|I| \sim |J|$. For this purpose we will derive a bound on the blockwise rank if an accuracy $\varepsilon > 0$ for the approximation error

$$\|A - A_{\mathscr{H}}\|_F < \varepsilon \|A\|_F$$

is prescribed. Due to the properties of the Frobenius norm, the previous estimate is implied by the condition

$$\|A_b - S_k\|_F < \varepsilon \|A_b\|_F \tag{3.69}$$

on each block $b = t \times s \in P$, where $A_{\mathscr{H}} \in \mathscr{H}(T_{I\times J}, k)$ is defined by $(A_{\mathscr{H}})_b = S_k$ if b is admissible and $(A_{\mathscr{H}})_b = A_b$ else.

Since the error estimates (3.63) and (3.68) depend on the choice of the approximation system Ξ, we cannot reveal the constants appearing in these estimates. Since they will enter the overall complexity through the logarithm anyhow, it will be assumed that they are bounded independently of $|t|$, $|s|$, and k. For the convergence speed of ACA we can apply the results of Sect. 3.3 to the expression in (3.61). There it was proved that an asymptotically smooth kernel κ on a pair of domains satisfying (3.35) can be approximated by polynomials of order p with accuracy η^p. We remark, however, that depending on the kernel and on the computational domain usually a faster convergence can be expected. For computational domains Ω that are $(d-1)$-dimensional manifolds one has for instance that $k = \dim \Pi_p^{d-1} \sim p^{d-1}$. For simplicity we therefore assume that for each admissible block $A_b \in \mathbb{R}^{t\times s}$ the approximant S_k, $k = \dim \Pi_p^d \sim p^d$, satisfies

$$\|A_b - S_k\|_F \le c\, \eta^p \|A_b\|_F.$$

If a block does not satisfy (3.35), then its original entries are stored. For

$$p \ge \log_{1/\eta} c/\varepsilon$$

we obtain that (3.69) is valid. Since $k \sim p^d$ and since the complexity on each block $t \times s$ is of the order $k^2(|t| + |s|)$, the overall complexity for generating the approximant $A_{\mathscr{H}} \in \mathscr{H}(T_{I\times J}, k)$ is $n \log n |\log \varepsilon|^{2d}$, where $n := |I| \sim |J|$; see (1.36). The complexities for storing $A_{\mathscr{H}}$ and multiplying it by a vector, once it has been generated, result from the complexity of $\mathscr{H}$-matrices in Chap. 2 and are both $n \log n |\log \varepsilon|^d$.

3.4.5 Numerical Experiments

In order to show that the proposed techniques work reliably and efficient, in the following numerical experiments we will apply ACA to complex and nonsmooth geometries. Furthermore, the results of applying ACA to the Helmholtz equation will be reported. As a third numerical example, we consider mixed boundary value problems of the Lamé equations.

3.4.5.1 Complex Geometries

From Example 3.38 it could be seen that ACA may run into trouble if not enough attention is paid to the choice of pivoting rows. This problem was caused by the normal ν to Γ appearing in the kernel of the double-layer integral operator and nonsmooth boundaries. In order to show that a thorough implementation (see Sect. 3.4.3) of ACA can handle nonsmooth geometries, we consider the Dirichlet boundary value problem

$$\begin{aligned} -\Delta u &= 0 \quad \text{in } \Omega, \\ u &= g \quad \text{on } \Gamma, \end{aligned}$$

where Γ is the surface shown in Fig. 3.4, which is by courtesy of Fraunhofer ITWM, Kaiserslautern. From (3.13) we obtain the following boundary integral equation for

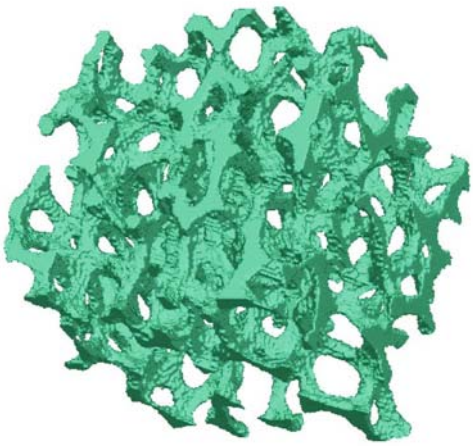


Fig. 3.4 Surface mesh of a foam with $n = 494\,616$ degrees of freedom.

the unknown function $t := \partial_\nu u$:

$$\mathscr{V}t = (\frac{1}{2}\mathscr{I} + \mathscr{K})g \quad \text{on } \Gamma.$$

A Galerkin discretization with piecewise linears φ_j, $j = 1,\ldots,n$, and piecewise constants ψ_i, $i = 1,\ldots,n'$, leads to the following linear system of equations

$$Vx = b, \quad b = (\frac{1}{2}M + K)\tilde{g}, \tag{3.70}$$

where for $i = 1,\ldots,n'$ and $j = 1,\ldots,n$

$$V_{ij} = (\mathscr{V}\psi_j, \psi_i)_{L^2(\Gamma)}, \quad K_{ij} = (\mathscr{K}\varphi_j, \psi_i)_{L^2(\Gamma)}, \quad \text{and} \quad M_{ij} = (\varphi_j, \psi_i)_{L^2(\Gamma)}.$$

Furthermore, $\tilde{g} \in \mathbb{R}^n$ is the vector minimizing

$$\|g - \sum_{j=1}^{n} \tilde{g}_j \varphi_j\|_{L^2(\Gamma)}.$$

The solution x of (3.70) defines an approximation $t_h := \sum_{i=1}^{n'} x_i \psi_i$ of t.

In the first set of numerical tests we approximate the matrices V and K of the single and double-layer operator using ACA (see Algorithm 3.1) with relative accuracy $\varepsilon = 1_{10}{-}4$. Additionally, the techniques from Sect. 2.6 are used to improve the compression rate. In all tests a minimal blocksize $n_{\min} = 15$ was chosen.

Since $\mathscr{V}$ is coercive, its Galerkin stiffness matrix V is symmetric positive definite. Hence, in contrast to the $\mathscr{H}$-matrix approximant $K_{\mathscr{H}}$ to K, we may generate only the upper triangular part of the approximant $V_{\mathscr{H}}$ to V. The positivity is preserved under sufficiently small perturbations such as the approximation error caused by ACA. Table 3.1 shows the time needed for building $V_{\mathscr{H}}$ and its memory consumption for two discretizations of Γ and different admissibility parameters η. Additionally, the ratio of the amount of storage required by the compressed and the standard representation can be found in the columns labeled "ratio". Table 3.2 contains the respective values for $K_{\mathscr{H}}$. We observe that increasing the number of degrees of freedom by

Table 3.1 Approximation results for V and $\varepsilon = 1_{10}{-}4$.

	$n = 28\,968$			$n = 115\,872$		
η	time	MB	ratio	time	MB	ratio
0.8	316s	259	8.1%	1 567s	1 264	2.5%
1.0	253s	204	6.4%	1 251s	995	1.9%
1.2	217s	173	5.4%	1 208s	967	1.9%
1.4	208s	162	5.1%	2 812s	2 513	4.9%

Table 3.2 Approximation results for K and $\varepsilon = 1_{10}{-}4$.

	$n = 28\,968$			$n = 115\,872$		
η	time	MB	ratio	time	MB	ratio
0.8	2 334s	543	17.4%	11 651s	2 789	5.5%
1.0	1 943s	443	14.2%	9 517s	2 264	4.4%
1.2	1 711s	386	12.3%	8 788s	2 119	4.2%
1.4	2 001s	475	15.2%	21 260s	6 222	12.2%

a factor of 4 leads to an almost five times larger amount of storage. The required CPU time behaves similarly. The optimal choice for η seems to be 1.2. For the computation of the singular integrals we used O. Steinbach's semi-analytic quadrature routines OSTBEM. Comparing the ratios of the required amount of storage and

the CPU time, one observes that the computation of each entry of the double-layer matrix requires about 3.5 times more CPU time than the single-layer entries require. This is due to the fact that for the approximation of g piecewise linears are used.

The CPU time for recompressing $V_{\mathscr{H}}$ to the same accuracy $\varepsilon = 1_{10}-4$ with the algebraic technique from Sect. 2.6 and the memory consumption of the resulting approximant are presented in Table 3.3. Apparently, the recompression technique

Table 3.3 Coarsening $V_{\mathscr{H}}$ with $\varepsilon = 1_{10}-4$.

	$n = 28\,968$			$n = 115\,872$		
η	time	MB	ratio	time	MB	ratio
0.8	25s	152	4.7%	116s	758	1.5%
1.0	19s	152	4.7%	83s	758	1.5%
1.2	13s	152	4.7%	53s	876	1.7%
1.4	10s	155	4.9%	48s	2411	4.7%

significantly reduces the storage requirements for small η, i.e., for fine partitions. The partition for $\eta = 1.2$ seems to be almost optimal such that the recompression does not lead to significant improvements. The choice $\eta = 1.4$ results in a partition which is too coarse for the recompression procedure to have its effects. Its application, however, is useful in any case, because the computational effort can be neglected compared with the construction of the matrix.

In the following tests we compare the accuracy of the computed solution $\tilde{t}_h$ with

$$t(x) := \partial_{\nu_x} S(x-y_0) = \frac{1}{4\pi} \frac{\nu_x \cdot (y_0 - x)}{\|x-y_0\|^3},$$

which is the Neumann data of the exact solution $u(x) = S(x-y_0)$ for the right-hand side $g_D(x) := S(x-y_0)$, $x \in \Gamma$. The point y_0 is chosen outside of $\overline{\Omega}$. Hence, we are able to compute the approximate L^2-error

$$\left(\sum_{i=1,\ldots,n} \mu(\pi_i) \, |t(m_i) - t_i)|^2 \right)^{1/2}$$

as measure for the error of the solution. Here, m_i denotes the centroid of the ith triangle π_i. The results for various approximation accuracies ε and the two discretizations of the boundary from Fig. 3.4 are presented in Table 3.4. For these tests $\eta = 1.2$ was chosen. The approximation accuracy ε has to be chosen relatively high in order to be able to observe the convergence of t_h against t. One of the reasons for this is that the different discretizations of Γ are not just refinements of an initial discretization, the topology of the discretizations may even differ. Hence, it happens that the accuracy of $\tilde{t}_h$ even deteriorates on the finer grid for larger ε.

In the previous table we have presented only the accuracy of the solution. Its preconditioned iterative or fast direct solution will be considered in Sect. 3.6.

Table 3.4 Solution error $\|t-\tilde{t}_h\|_2$ for different ε.

ε	$n=28968$	$n=115872$
$1_{10}-4$	$3.2_{10}-2$	$6.7_{10}-2$
$1_{10}-5$	$3.3_{10}-3$	$4.8_{10}-3$
$1_{10}-6$	$2.4_{10}-3$	$1.5_{10}-3$

3.4.5.2 Helmholtz' Equation

Boundary integral formulations can be derived and are particularly useful for Helmholtz' equation

$$\begin{aligned} -\Delta u - \omega^2 u &= 0 \quad \text{in } \Omega^c, \\ u &= 1 \quad \text{on } \partial\Omega, \end{aligned}$$

where $\Omega \subset \mathbb{R}^d$ is a bounded domain and $\Omega^c = \mathbb{R}^d \setminus \overline{\Omega}$; see [45, 55]. From Lemma 3.5 it can be seen that the singularity function of the Helmholtz operator $-\Delta - \omega^2$ is asymptotically smooth such that ACA is applicable. However, the constant γ appearing in the definition of asymptotic smoothness (3.18) (and thus in the error estimates of all degenerate kernel approximations and also of ACA) contains the factor

$$c_{\mathscr{L}} = \left(72 \cdot 73 + 2\omega^2 (\operatorname{diam}\Omega)^2\right)^{1/2}$$

due to Lemma 3.3 and Lemma 3.5, which shows that the required rank k scales linearly with respect to ω for large wave numbers ω. Additionally, the mesh size h has to be chosen such that $\omega h \ll 1$ when discretizing $\partial\Omega$; see for instance [102]. As a consequence, $k \sim h^{-1} \sim n^{1/(d-1)}$ and methods based on low-rank approximants will not be able to achieve logarithmic-linear complexity for large ω. $\mathscr{H}$-matrices will therefore have complexity $n^{d/(d-1)}$; see also [216, 217] for multipole algorithms of complexity $n^{3/2}$ and $n^{4/3}$. In the Table 3.5 we compare the required rank for a prescribed accuracy $\varepsilon = 1_{10}-4$ of ACA applied to an admissible sub-block with the low-rank approximant resulting from the SVD for increasing wave numbers ω. Apparently, ACA gives an almost optimal approximation, while the optimum rank

Table 3.5 Low-rank approximation for increasing wave numbers.

		SVD		ACA	
ω	matrix size	k	time	k	time
20	8×16	8	0.00s	8	0.00s
25	15×31	10	0.00s	13	0.00s
50	125×250	19	0.05s	26	0.00s
100	1000×2000	37	32.51s	42	0.05s
200	8000×16000	–	–	84	2.13s
400	64000×128000	–	–	175	100.82s

revealed by the SVD grows linearly with ω. If we want to guarantee that $\omega h \sim 1$, then the matrix dimensions $m = (\omega/10)^3$ and $n = 2m$ will grow cubically with respect to ω. As a consequence, the SVD, which requires to compute the whole matrix beforehand, cannot be applied for $\omega \geq 200$ due to memory limitations. As a conclusion, low-rank techniques will not lead to optimal complexity. ACA provides an easy and efficient means to compute an optimal approximation in the class of low-rank methods. Recently, new multilevel techniques (see [217, 88, 207, 74, 5, 12]) have been introduced, which achieve the desired logarithmic-linear asymptotic complexity.

3.4.5.3 Problems from Linear Elasticity

In the rest of this section we apply ACA to boundary element matrices arising from linear elasticity in three spatial dimensions. The panel clustering method has been applied to the Lamé equations in [143], fast multipole methods where used in [194].

We consider an elastic body $\Omega \subset \mathbb{R}^3$ with Lipschitz boundary Γ. Suppose that a displacement of the body g_D is given on $\Gamma_D \subset \Gamma$ and some force g_N is applied to $\Gamma_N \subset \Gamma = \overline{\Gamma}_D \cup \overline{\Gamma}_N$. To find displacements and forces inside Ω, we set up a mixed boundary value problem (cf. (3.1)) for the Lamé system:

$$-\mu\vec{\Delta}u - (\lambda+\mu)\nabla\operatorname{div} u = 0, \quad x \in \Omega, \tag{3.71a}$$

$$u = g_D, \quad x \in \Gamma_D, \tag{3.71b}$$

$$t := Tu = g_N, \quad x \in \Gamma_N. \tag{3.71c}$$

Here, $u : \mathbb{R}^3 \to \mathbb{R}^3$ is the displacement vector and

$$Tu = \lambda \operatorname{div} u \cdot \nu + 2\mu\, \nu \cdot \nabla u + \mu\, \nu \times \operatorname{curl} u \tag{3.72}$$

is the traction on the boundary. The Lamé constants λ and μ relate to the elasticity module $E > 0$ and the Poisson ratio $r \in (0, \frac{1}{2})$ of the material as follows

$$\lambda = \frac{Er}{(1+\mu)(1-2r)}, \quad \mu = \frac{E}{2(1+r)}. \tag{3.73}$$

Problem (3.71) admits the following symmetric boundary integral formulation; cf. (3.13).

$$\mathscr{V}\tilde{t} - \mathscr{K}\tilde{u} = \left(\frac{1}{2}\mathscr{I} + \mathscr{K}\right)\tilde{g}_D - \mathscr{V}\tilde{g}_N \quad \text{on } \Gamma_D, \tag{3.74a}$$

$$\mathscr{K}'\tilde{t} + \mathscr{D}\tilde{u} = \left(\frac{1}{2}\mathscr{I} - \mathscr{K}'\right)\tilde{g}_N - \mathscr{D}\tilde{g}_D \quad \text{on } \Gamma_N, \tag{3.74b}$$

where $\tilde{u} = u - \tilde{g}_D$, $\tilde{t} = t - \tilde{g}_N$, $\tilde{g}_N$, and $\tilde{g}_D$ are extensions of g_N and g_D to the whole boundary Γ. Furthermore,

$$
\begin{aligned}
(\mathscr{V}t)(x) &= \int_\Gamma [S(x-y)]^T t(y)\,\mathrm{d}s_y,\\
(\mathscr{K}u)(x) &= \int_\Gamma [T_y^* S(x-y)]^T u(y)\,\mathrm{d}s_y,\\
(\mathscr{K}'t)(x) &= \int_\Gamma [T_x S(x-y)]^T t(y)\,\mathrm{d}s_y,\\
(\mathscr{D}u)(x) &= -T_x \int_\Gamma [T_y^* S(x-y)]^T u(y)\,\mathrm{d}s_y
\end{aligned}
$$

are the single-layer, double-layer, adjoint double-layer potentials and a hypersingular operator, respectively. In the previous expression

$$
S_{ij}(x,y) = \frac{1+r}{8\pi E(1-r)}\left[(3-4r)\frac{\delta_{ij}}{\|x-y\|} + \frac{(x_i-y_i)(x_j-y_j)}{\|x-y\|^3}\right] \tag{3.75}
$$

for $i,j=1,2,3$ denotes the singularity function. For details we refer to [236, 70, 237] and [242].

Although we could apply ACA directly to the discrete operators, it is more efficient to exploit their relation to the respective operators of the Laplacian. As it was shown in [167] (see also [242]), the action of $\mathscr{K}$ and $\mathscr{K}'$ can be expressed in terms of $\mathscr{V}$, $\mathscr{V}_\Delta$, $\mathscr{K}_\Delta$, and $\mathscr{R}$

$$
(\mathscr{K}u,t) = 2\mu(\mathscr{V}\mathscr{R}u,t) + (\mathscr{K}_\Delta u,t) - (\mathscr{V}_\Delta \mathscr{R}u,t), \tag{3.76a}
$$

$$
(\mathscr{K}'t,v) = 2\mu(\mathscr{V}t,\mathscr{R}v) + (\mathscr{K}'_\Delta t,v) - (\mathscr{V}_\Delta t,\mathscr{R}v), \tag{3.76b}
$$

where $\mathscr{V}_\Delta$, $\mathscr{K}_\Delta$, and $\mathscr{K}'_\Delta$ are the corresponding operators for the Laplacian and

$$
\mathscr{R}_{ij} = \nu_j\partial_i - \nu_i\partial_j, \quad i,j=1,2,3.
$$

The action of the hypersingular operator can also be expressed in terms of $\mathscr{V}$, $\mathscr{V}_\Delta$, and $\mathscr{R}$ (see [167], [242]):

$$
\begin{aligned}
(\mathscr{D}u,v) = &\int_\Gamma\int_\Gamma \frac{\mu}{4\pi\|x-y\|}\left(\sum_{k=1}^3 \frac{\partial u}{\partial S_k}(y)\frac{\partial v}{\partial S_k}(x)\right)\\
&+ (\mathscr{R}v)^T(x)\left(\frac{\mu}{2\pi\|x-y\|}\mathscr{I} - 4\mu^2 S(x-y)\right)(\mathscr{R}u)(y)\\
&+ \sum_{i,j,k=1}^3 (\mathscr{R}_{kj}v_i)(x)\frac{1}{\|x-y\|}(\mathscr{R}_{ki}u_j)(y)\,\mathrm{d}s_x\,\mathrm{d}s_y.
\end{aligned} \tag{3.77}
$$

Here,

$$
\frac{\partial}{\partial S_1} = \mathscr{R}_{32}, \quad \frac{\partial}{\partial S_2} = \mathscr{R}_{13}, \quad \text{and} \quad \frac{\partial}{\partial S_3} = \mathscr{R}_{21}.
$$

To see how useful the above relations are, we proceed with the variational formulation of the problem. Multiplying equations (3.74) by test functions $\tau \in \tilde{H}^{-1/2}(\Gamma_D)$ and $v \in \tilde{H}^{1/2}(\Gamma_N)$, respectively, and integrating the resulting equations over the

surface gives

$$a(\tilde{u},\tilde{t};v,\tau) = f(v,\tau) \quad \text{for all } (v,\tau) \in \tilde{H}^{1/2}(\Gamma_N) \times \tilde{H}^{-1/2}(\Gamma_D),$$

where

$$a(\tilde{u},\tilde{t};v,\tau) = (\mathscr{V}\tilde{t},\tau)_{L^2(\Gamma_D)} - (\mathscr{K}\tilde{t},\tau)_{L^2(\Gamma_D)} + (\mathscr{K}'\tilde{t},v)_{L^2(\Gamma_N)} + (\mathscr{D}\tilde{u},v)_{L^2(\Gamma_N)}$$

and

$$\begin{aligned} f(v,\tau) = &\left(\left(\frac{1}{2}\mathscr{I} + \mathscr{K}\right)\tilde{g}_D,\tau\right)_{L^2(\Gamma_D)} - (\mathscr{V}\tilde{g}_N,\tau)_{L^2(\Gamma_D)} \\ &+ \left(\left(\frac{1}{2}\mathscr{I} - \mathscr{K}'\right)\tilde{g}_N,v\right)_{L^2(\Gamma_N)} - (\mathscr{D}\tilde{g}_D,v)_{L^2(\Gamma_N)}. \end{aligned}$$

We discretize the spaces $H^{-1/2}(\Gamma) \approx \operatorname{span}\{\psi_i\}_{i=1}^{n'}$ and $H^{1/2}(\Gamma) \approx \operatorname{span}\{\varphi_j\}_{j=1}^{n}$ looking for the solution of the form

$$u_h(x) = \sum_{j=1}^{n} \begin{bmatrix} u_{x,j} \\ u_{y,j} \\ u_{z,j} \end{bmatrix} \varphi_j(x), \quad t_h(x) = \sum_{i=1}^{n'} \begin{bmatrix} t_{x,i} \\ t_{y,i} \\ t_{z,i} \end{bmatrix} \psi_i(x).$$

This gives the following system (cf. (3.16)) of linear algebraic equations for the partially unknown coefficient vectors $u = [u_{x,j}, u_{y,j}, u_{z,j}]_{j=1}^{n_D}$ and $t = [t_{x,i}, t_{y,i}, t_{z,i}]_{i=1}^{n'_N}$

$$\begin{bmatrix} -V & K \\ K^T & D \end{bmatrix} \begin{bmatrix} t \\ u \end{bmatrix} = \begin{bmatrix} V & -\frac{1}{2}M - K \\ \frac{1}{2}M - K^T & -D \end{bmatrix} \begin{bmatrix} \tilde{g}_N \\ \tilde{g}_D \end{bmatrix} =: \begin{bmatrix} f_N \\ f_D \end{bmatrix}. \tag{3.78}$$

The entries of the above matrices are

$$V_{k\ell} = (\mathscr{V}\psi_\ell, \psi_k)_{L^2}, \quad K_{kj} = (\mathscr{K}\varphi_j, \psi_k)_{L^2}, \quad D_{ij} = (\mathscr{D}\varphi_j, \varphi_i)_{L^2},$$

where $k,\ell = 1,\dots,n_D$ and $i,j = 1,\dots,n'_N$. In order to be able to deal with a large number of unknown coefficients, the discrete system (3.78) is solved using a Krylov subspace method. To use this kind of iterative solver, we only need to provide the multiplication of the coefficient matrix by a vector. This is done using representations (3.76) and (3.77), thereby avoiding explicit computations of the entries of D and K. The only dense matrices that need to be stored are V, V_Δ, and K_Δ. By reordering the unknown coefficients such that the components are separated

$$\begin{aligned} \{u_{x,j}, u_{y,j}, u_{z,j}\}_{j=1}^{n} &\to \{\{u_{x,j}\}_{j=1}^{n}, \{u_{y,j}\}_{j=1}^{n}, \{u_{z,j}\}_{j=1}^{n}\}, \\ \{t_{x,i}, t_{y,i}, t_{z,i}\}_{i=1}^{n'} &\to \{\{t_{x,i}\}_{i=1}^{n'}, \{t_{y,i}\}_{i=1}^{n'}, \{t_{z,i}\}_{i=1}^{n'}\}, \end{aligned}$$

the matrix V can be decomposed as

$$V = c_1 \begin{bmatrix} V_\Delta & 0 & 0 \\ 0 & V_\Delta & 0 \\ 0 & 0 & V_\Delta \end{bmatrix} + c_2 \begin{bmatrix} V_1 & V_2 & V_3 \\ V_2 & V_4 & V_5 \\ V_3 & V_5 & V_6 \end{bmatrix},$$

where V_i, $i = 1,\ldots,6$, are the matrices corresponding to the entries of the second term in (3.75). Thus, for the solution of the system (3.78) we need to store seven Hermitian $n_D \times n_D$ matrices to represent V and one $n_D \times n'_N$ matrix K_Δ. These matrices are composed using ACA on admissible blocks.

We first compute the matrices for the geometry depicted in Fig. 3.5. Nodes on the left face of the beam are fixed ($u = 0$) and a unit vertical force $t = (0,0,-1)$ is applied to each triangle of the right face. The mesh is uniformly refined to test the performance on matrices of different sizes. The compression results for the matrix

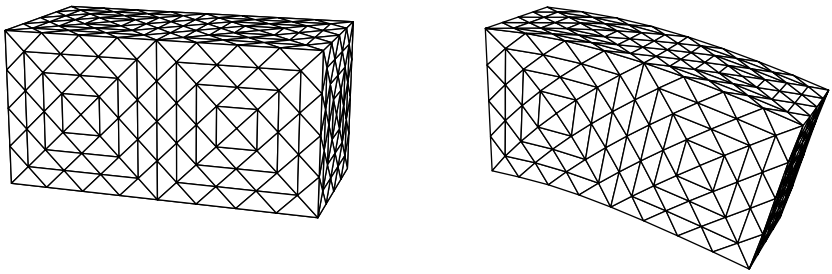


Fig. 3.5 Test domain before and after the elastic deformation.

V are shown in Table 3.6 for fixed $\varepsilon = 1_{10}{-}5$. As expected, the compression ratio (column 3) improves for increasing n. Furthermore, we observe a decay of the memory increment (column 4) from one problem size to the next. This reflects the almost linear storage requirement of the matrix. The amount of computed matrix entries in millions is shown in column 5 of the table. Column 6 shows that the compression procedure uses only a small portion of the original matrix entries to build a blockwise low-rank approximant. Table 3.7 reveals the dependence of the compression

Table 3.6 Compression and number of computed entries of V ($\varepsilon = 1_{10}{-}5$).

	memory			CPU time		
n	MB	ratio	incr.	entries	ratio	incr.
640	9.50	0.338	–	2.7	0.738	–
1 280	30.87	0.274	3.250	8.8	0.600	3.250
2 560	90.37	0.201	2.928	25.8	0.439	2.930
5 120	239.49	0.133	2.650	64.8	0.275	2.510
10 240	562.58	0.078	2.349	141.6	0.150	2.182
20 480	1 291.78	0.045	2.296	358.5	0.099	2.531
40 960	2 624.02	0.023	2.031	820.4	0.054	2.288

rate of the matrix V on the approximation accuracy ε. For these tests we have chosen $n = 20480$ such that V has dimension $3n \times 3n$. As predicted, the approximation

Table 3.7 Compression of V for various ε ($n = 20480$).

ε	MB	ratio
$1_{10}-2$	458.01	0.016
$1_{10}-3$	633.51	0.022
$1_{10}-4$	848.91	0.030
$1_{10}-5$	1 291.78	0.045

accuracy ε enters the complexity only through the logarithm.

The next tests are performed on the geometry shown in Fig. 3.6. Nodes on the bottom face of the body are fixed ($u = 0$) and a unit vertical force $t = (0,0,1)$ is applied to each triangle of the top face. The performance results are collected in Table 3.8. We observe a behavior similar to the behavior in the previous tests.

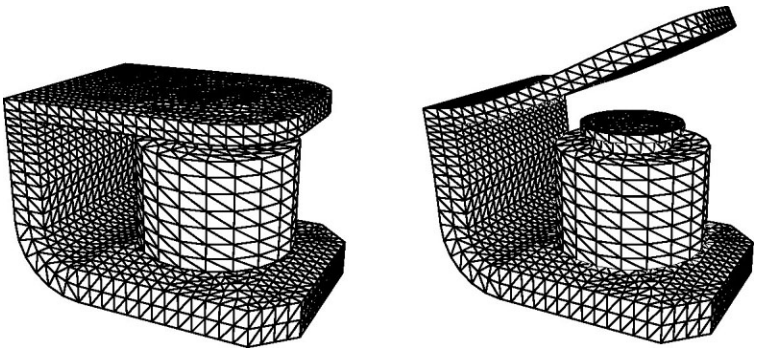


Fig. 3.6 Test domain before and after the elastic deformation.

Table 3.8 Compression results for the second test geometry ($\varepsilon = 1_{10}-4$).

	memory			CPU time		
n	MB	ratio	incr.	entries	ratio	incr.
4 944	278.26	0.1658	–	61	0.275	–
19 776	1 791.83	0.0667	6.439	361	0.102	5.960
79 104	9 499.65	0.0221	5.302	1 786	0.032	4.951

3.4.6 Parallelization of ACA

Although an $\mathscr{H}$-matrix approximant can be computed using ACA with $\mathcal{O}(k^2 n \log n)$ operations, building the matrix is still the most time-consuming part of the computation. Hence, it would be beneficial to the execution time if the construction of the approximant could be done in parallel; cf. [202]. Therefore, in this section the parallel construction of the $\mathscr{H}$-matrix approximant is discussed. Algorithms for shared and for distributed memory are introduced which show an optimal parallel speedup; cf. [29]. The basis for these parallel algorithms is the following sequential algorithm for the approximation of $A \in \mathbb{R}^{n \times n}$.

```
for all b ∈ P do
    if b is admissible then
        create a low-rank approximant of A_b using ACA;
    else
        create the dense matrix A_b;
endfor;
```

The result is either a low-rank matrix if the block satisfies (3.35) or a dense matrix if it does not satisfy the admissibility criterion. Since in both cases the computation on each block is independent from the other blocks, the construction process has a natural parallelism, which promises optimal speedups if a reasonable scheduling is available. An immediate approach is to subdivide the matrix into $p = 4^q$, $q \in \mathbb{N}$, parts using the partition on the qth level of the block cluster tree, where p is the number of processors. This approach, however, will not lead to competitive speedups since the costs of sub-blocks on the diagonal, for instance, are much higher then those of off-diagonal sub-blocks. In addition, this block structure does not account for the parallel matrix-vector multiplication introduced in Sect. 2.3.

To achieve a good parallel speedup, usually prior knowledge of the amount of work per matrix block is needed for balancing the load among the processors. Unfortunately, this information is only accessible if the low-rank matrix blocks are computed with a fixed rank. If, however, the stopping criterion (3.58), which guarantees a prescribed accuracy, is used in the ACA algorithm, the rank is chosen adaptively and may hence vary from block to block. Therefore, the actual costs per matrix block are unknown.

An alternative to a cost-related load balancing algorithm is **list scheduling**. This scheduling algorithm works by assigning the next not yet executed job to the first idle processor, independently of the amount of work associated with the job. A version of the above described $\mathscr{H}$-matrix approximation algorithm utilizing list scheduling is shown in Algorithm 3.2.

```
for all b ∈ P do
    let 0 ≤ i < p be the number of the first idle processor;
    if b is admissible then
        create a low-rank approximant of A_b using ACA on processor i;
    else
        create the dense matrix A_b on processor i;
endfor;
```

Algorithm 3.2: List scheduling.

List scheduling does not produce an optimal scheduling, but one can prove the following estimate; see [113].

Lemma 3.41. *Let $t(p)$ be the time for executing n jobs on p processors using list scheduling and let $t_{\min}(p)$ be the minimal time needed for running n jobs on p processors. Then it holds that*

$$t(p) \leq \left(2 - \frac{1}{p}\right) t_{\min}(p). \tag{3.79}$$

If the costs are known a-priori, e.g., if a constant rank approximation is generated, one is able to use a refined version of list scheduling, namely **longest process time (LPT) scheduling**. Instead of assigning the matrix blocks randomly to the processors, they are ordered according to their costs, starting from the most expensive one.

```
V := P;
while V ≠ ∅ do
    choose v ∈ V with maximum costs; V := V \ {v};
    let 0 ≤ i < p be the first idle processor;
    if v is admissible then
        create a rank-k approximant of A_b using ACA on processor i;
    else
        create the dense matrix A_b on processor i;
endfor;
```

Algorithm 3.3: LPT scheduling.

Compared with list scheduling, the result of this ordering is an improved approximation of an optimal scheduling; see [113].

Lemma 3.42. *Let $t(p)$ be the time for executing n jobs on p processors using LPT scheduling and let $t_{\min}(p)$ be as in Lemma 3.41. Then it holds that*

$$t(p) \leq \left(\frac{4}{3} - \frac{1}{3p}\right) t_{\min}(p). \tag{3.80}$$

Since according to Lemma 3.41 and Lemma 3.42 the costs differ from an optimal scheduling only by a constant factor, the numerical effort for building the approximant is of order $p^{-1}k^2 n \log n$.

3.4.6.1 Shared Memory Systems

Using list scheduling for building an $\mathscr{H}$-matrix, the result (3.79) ensures an efficient algorithm with a parallel speedup of at least $p/2$. Furthermore, numerical experiments indicate an almost optimal parallel behavior of the resulting algorithm independently of the type of approximation (fixed or adaptive rank). In the case of shared memory systems we will therefore restrict ourselves to this scheduling method.

A widely used mechanism for parallel computations on shared memory systems are *threads* which are independent, parallel execution paths in a single task. Because all threads are part of the same task, they share the same address space which eliminates the need for communication between individual processors. A standard interface to threads on most computer systems are *POSIX threads*; see [57]. Unfortunately, the usage of the POSIX thread interface can be complicated and often distracts from the real problem to solve. Furthermore, the creation of threads comes with some costs, especially if several hundreds of threads are used. The latter situation occurs in a modified version of Algorithm 3.2, where each submatrix is computed in a new thread:

```
for all b ∈ P do
    create a new thread t;
    if b is admissible then
        build a low-rank approximant of A_b using ACA in thread t;
    else
        build the dense matrix A_b in thread t;
endfor;
```

A convenient and efficient interface to POSIX threads is the threadpool library presented in [165].

Example 3.43. We apply the proposed methods to $\mathscr{H}$-matrices stemming from the Galerkin discretization of the single-layer potential operator $\mathscr{V} : H^{-1/2}(\Gamma) \to H^{1/2}(\Gamma)$ defined by

$$(\mathscr{V}u)(x) = \frac{1}{4\pi} \int_\Gamma \frac{u(y)}{\|x-y\|} \, \mathrm{d}s_y$$

on the surface Γ from Fig. 3.7.

A Galerkin discretization with piecewise constants ψ_i, $i = 1, \ldots, n$, leads to the matrix $V \in \mathbb{R}^{n \times n}$ with entries

$$V_{ij} = (\mathscr{V}\psi_j, \psi_i)_{L^2(\Gamma)}, \quad i, j = 1, \ldots, n,$$

which is symmetric since $\mathscr{V}$ is self-adjoint with respect to $(\cdot,\cdot)_{L^2(\Gamma)}$. Therefore, it is only necessary to approximate the upper triangular part of V. For the numerical tests of the shared memory algorithms an HP9000, PA-RISC with 875 MHz was used.

Table 3.9 shows the time and the corresponding parallel efficiency E (see Definition 2.9) in percent for building an $\mathscr{H}$-matrix using a fixed rank of $k = 10$ for different numbers of processors. For this purpose Algorithm 3.2 was used. Table 3.10

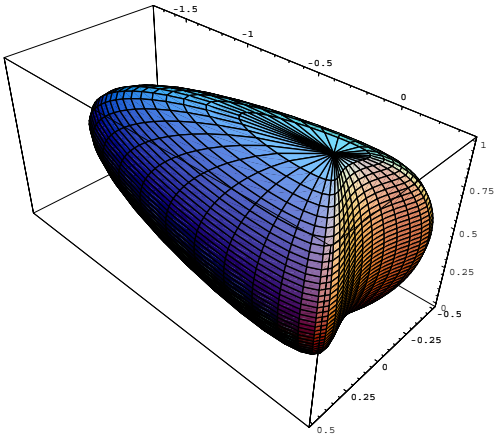


Fig. 3.7 The test surface.

Table 3.9 Parallel ACA with $k = 10$.

	$p=1$	$p=4$		$p=8$		$p=12$		$p=16$		size
n	time	time	E	time	E	time	E	time	E	in MB
3 968	117s	29s	99%	14s	99%	9s	99%	7s	99%	33
7 920	320s	80s	99%	40s	99%	26s	99%	20s	99%	83
19 320	1 042s	261s	99%	130s	99%	87s	99%	65s	99%	258
43 680	2 904s	727s	99%	364s	99%	243s	99%	182s	99%	706
89 400	6 964s	1 747s	99%	875s	99%	582s	99%	437s	99%	1 670
184 040	16 694s	4 179s	99%	2 081s	100%	1 391s	99%	1 049s	99%	3 980

contains the execution times for building the $\mathscr{H}$-matrix using a fixed approximation accuracy $\varepsilon = 1_{10}{-}4$.

Table 3.10 Parallel ACA with $\varepsilon = 1_{10}{-}4$.

	$p=1$	$p=4$		$p=8$		$p=12$		$p=16$		size
n	time	time	E	time	E	time	E	time	E	in MB
3 968	101s	25s	99%	12s	99%	8s	99%	6s	99%	46
7 920	269s	67s	99%	33s	99%	22s	99%	16s	99%	88
19 320	826s	206s	99%	103s	99%	69s	99%	51s	99%	226
43 680	2 258s	566s	99%	283s	99%	192s	99%	142s	99%	577
89 400	5 362s	1 347s	99%	675s	99%	448s	99%	337s	99%	1 326
184 040	12 741s	3 185s	100%	1 596s	99%	1 059s	100%	796s	100%	3 079

The algorithm shows a very high parallel efficiency even for small problem sizes, which indicates the low overhead resulting from the usage of the threadpool and the almost optimal load balancing produced by list scheduling.

3.4.6.2 Distributed Memory Systems

In the first part of this section we consider the case of constant rank approximations. In order to be able to apply LPT scheduling, the costs per matrix block have to be known. We define the following cost functional for each matrix block $b = t \times s \in P$ by

$$c_k(t,s) = \begin{cases} |t| \cdot |s|, & t \times s \text{ is admissible,} \\ k^2 \cdot (|t| + |s|), & \text{else.} \end{cases}$$

Algorithm 3.3 associates blocks to processors during the computation. Although this *online* scheduling can be implemented also for distributed memory systems, e.g., by a master-slave model, this involves communication which should be avoided. Hence, in the case of a distributed memory an *offline* version is favorable; i.e., the load balancing has to be done in advance. The offline version of Algorithm 3.3 is shown in Algorithm 3.4.

```
procedure LPT(P)
    V := P; C_i := 0 for 0 ≤ i < p;
    while V ≠ ∅ do
        let v = argmax_{b∈V} c_k(b);
        let i = argmax_{0≤j<p} C_j;
        assign v ∈ V to processor i;
        C_i := C_i + c_k(v); V := V \ {v};
    endwhile;
end;
```

Algorithm 3.4: Offline LPT scheduling.

The sum of the costs associated with each processor $0 \le i < p$ are stored in C_i. Hence, processor i, for which $C_i = \min_{0 \le j < p} C_j$, is the first idle processor to receive the next block.

As mentioned above, a cost functional for load balancing cannot be defined in the case of an adaptive rank approximation. In order to be able to still use offline scheduling, we assume that the ranks of the low-rank matrices are bounded by some number $k_{\max}$. Then an average rank k_{avg} for these submatrices exists which can be used in the same way as a constant rank. Estimate (3.80), however, will not hold in general, because only an approximation of the real costs per matrix block is used. Instead, the algorithm can be regarded as list scheduling and hence (3.79) is valid. Therefore, the resulting distribution still approximates an optimal load balancing up to a constant factor. Since the costs for building and for storing each block have the same complexity (see Remark 3.31), the amount of storage on each processor needed to hold the $\mathscr{H}$-matrix approximant is of the order $p^{-1} kn \log n$.

In Sect. 2.3 a scheduling algorithm based on sequence partitioning with space-filling curves was presented to balance the $\mathscr{H}$-matrix-vector multiplication among the processors. Although this scheduling is not optimal for the construction using ACA, it can also be used to distribute the blocks when building the $\mathscr{H}$-matrix approximant. Numerical experiments have shown that the advantage of LPT scheduling over sequence partitioning with space-filling curves can be neglected when

building the matrix. Hence, sequence partitioning should be preferred, especially if also the matrix-vector multiplication is to be executed in parallel, since then the redistribution of blocks can be avoided.

The following tests of the distributed memory algorithms were carried out on an AMD 900 MHz cluster. Due to memory restrictions, problems for large n could not be computed with a small number of processors p. The corresponding parallel efficiency for these problem sizes is therefore computed with respect to the smallest number of processors p' which was able to compute the problem; i.e.,

$$E(p) := \frac{p' \cdot t(p')}{p \cdot t(p)}.$$

The presented storage size in these cases is approximated by $p'N_{\mathrm{st}}$, where N_{st} denotes the memory consumption per processor on a system with p' CPUs. Table 3.11 shows the time for building the $\mathscr{H}$-matrix with a fixed rank of $k = 10$. The results

Table 3.11 Parallel application of ACA with $k = 10$.

	$p=1$	$p=4$		$p=8$		$p=12$		$p=16$		size
n	time	time	E	time	E	time	E	time	E	in MB
3 968	147s	37s	98%	18s	97%	13s	94%	10s	92%	29
7 920	404s	102s	99%	51s	97%	35s	94%	27s	92%	77
19 320	1 317s	334s	98%	166s	98%	119s	92%	87s	93%	245
43 680		929s		466s	*99%*	330s	*93%*	248s	*93%*	*712*
89 400				877s		605s	*96%*	467s	*94%*	*1 784*
184 040						1 430s		1 075s	*98%*	*4 892*

show that sequence partitioning in the case of constant rank approximations is highly efficient and also works very well for small problem sizes.

In Table 3.12 the results for a fixed accuracy of $\varepsilon = 1_{10}{-}4$ are reported. The average rank which is needed in this case was set to $k_{\mathrm{avg}} = 10$ with a maximal rank of $k_{\max} = 100$. Again, the $\mathscr{H}$-matrix could not be built for larger problem sizes and small p. Since the real costs per matrix block are only approximated, the parallel

Table 3.12 Parallel application of ACA for $\varepsilon = 1_{10}{-}4$.

	$p=1$	$p=4$		$p=8$		$p=12$		$p=16$		size
n	time	time	E	time	E	time	E	time	E	in MB
3 968	128s	33s	94%	18s	89%	12s	87%	9s	85%	26
7 920	338s	90s	93%	48s	88%	32s	86%	25s	84%	66
19 320	1 038s	272s	95%	143s	90%	96s	89%	74s	87%	197
43 680	2 845s	739s	96%	395s	90%	258s	91%	202s	87%	530
89 400				941s		628s	*100%*	480s	*98%*	*1 528*
184 040						1 450s		1 107s	*98%*	*3 864*

efficiency in the case of fixed accuracy is not as high as in the case of fixed rank approximations. Nevertheless, the scheduling still results in reasonable speedups.

3.5 A Recompression Technique for ACA (RACA)

The adaptive cross approximation method from Sect. 3.4 generates low-rank approximations UV^T, $U \in \mathbb{R}^{m\times k}$, $V \in \mathbb{R}^{n\times k}$, to suitable sub-blocks $A \in \mathbb{R}^{m\times n}$ of discrete integral formulations of elliptic boundary value problems. A characteristic property is that the approximation, which requires $k(m+n)$, $k \sim |\log \varepsilon|^*$, units of storage, is generated in an adaptive and purely algebraic manner using only few of the matrix entries. In this section (see [30]) we present a recompression technique which brings the required amount of storage down to sublinear order kk', where k' depends logarithmically on the accuracy of the approximation but is independent of the matrix size.

Although ACA generates approximants of high quality, the amount of storage required for an approximant can still be reduced. The reason for this is visible from the special structure of the approximation. In Lemma 3.32 we have pointed out that for the computed approximation it holds that

$$UV^T = A_{1:m,j_{1:k}} A_k^{-1} A_{i_{1:k},1:n}, \tag{3.81}$$

where $A_k := A_{i_{1:k},j_{1:k}}$. The representation (3.81) uses parts $A_{1:m,j_{1:k}}$ and $A_{i_{1:k},1:n}$ of the original matrix A for its approximation. Hence, the matrix entries of $A_{1:m,j_{1:k}}$ and $A_{i_{1:k},1:n}$ have the same smoothness properties as the original matrix A. The smoothness of the kernel function with respect to y was used by ACA. However, according to Lemma 3.5, the singularity function S is asymptotically smooth even with respect to both variables, i.e., for $x \in D_1$ and $y \in D_2$, where $D_1, D_2 \subset \Omega$, there are constants $c, \gamma_1, \gamma_2 > 0$ such that

$$|\partial_x^\alpha \partial_y^\beta S(x,y)| \le c\,|\alpha|!\,|\beta|!\,\gamma_1^{|\alpha|}\gamma_2^{|\beta|}\|x-y\|^{-|\alpha|-|\beta|}|S(x,y)| \tag{3.82}$$

for all $\alpha, \beta \in \mathbb{N}^d$. The smoothness with respect to x has not been exploited so far. Our aim in this section is to approximate the singularity function S as a function of x and as a function of y using Chebyshev polynomials, i.e., we will construct coefficient matrices $X_1, X_2 \in \mathbb{R}^{k'\times k}$, $k' \approx k$, such that for a given $\varepsilon > 0$

$$\|A_{1:m,j_{1:k}} - B_1 X_1\|_F \le \varepsilon \|A_{1:m,j_{1:k}}\|_F \quad \text{and} \tag{3.83a}$$

$$\|A^T_{i_{1:k},1:n} - B_2 X_2\|_F \le \varepsilon \|A_{i_{1:k},1:n}\|_F, \tag{3.83b}$$

where $B_1 \in \mathbb{R}^{m\times k'}$ and $B_2 \in \mathbb{R}^{n\times k'}$ are explicitly given matrices which are generated from evaluating Chebyshev polynomials and which do not depend on the matrix entries of A.

Note that our construction will not be based on explicit polynomial approximation of the kernel function since we can afford more advanced methods due to the fact that one of the dimensions of $A_{1:m,j_{1:k}}$ and $A_{i_{1:k},1:n}$ is k, which can be considered to be small. Our aim is to devise a method which preserves both the adaptivity of ACA and the property that only the matrix entries are used. The way we will achieve this is based on projecting $A_{1:m,j_{1:k}}$ and $A_{i_{1:k},1:n}$ to the bases B_1 and B_2, respectively. In total, this recompression generates uniform $\mathscr{H}$-matrices (see Sect. 2.11) from few of the original matrix entries. Notice that it is not required to develop arithmetic operations for uniform $\mathscr{H}$-matrices if approximate preconditioners are to be computed from the generated approximation. Since the recompression is based on ACA, one can generate an $\mathscr{H}$-matrix approximation of reduced accuracy as a byproduct and contruct a usual $\mathscr{H}$-matrix preconditioner from it.

For uniform $\mathscr{H}$-matrices it is necessary to store the coefficients of the projection together with the bases. The amount of storage for the coefficients is of the order $k|I|$, i.e., compared with $\mathscr{H}$-matrices the factor $\log|I|$ is saved. However, storing the bases still requires $\mathscr{O}(k|I|\log|I|)$ units of storage. In our construction the "basis matrices" B_1 and B_2 do not have to be stored. Only $X_1 \in \mathbb{R}^{k'\times k}$ and $X_2 \in \mathbb{R}^{k'\times k}$ will be stored and the matrices B_1 and B_2 will be recomputed every time they are used. Later it will be seen that for each $m \times n$ block the construction of B_1 and B_2 can be done with $k'm$ and $k'n$ operations, respectively. Hence, we will improve the overall asymptotic complexity to $\mathscr{O}(k|I|)$ units of storage due to (1.15). The matrix-vector multiplication with B_1X_1 takes $\mathscr{O}(k'(m+k))$ floating point operations. It will be seen that this additional effort does not change the asymptotic complexity of the matrix-vector multiplication. A slight increase of the actual run-time can be tolerated since the multiplication is computationally cheap while it is necessary to further reduce the storage requirements of ACA approximants.

In this section we will concentrate on a single sub-block $A \in \mathbb{R}^{m\times n}$ of an hierarchical matrix in $\mathbb{R}^{I\times J}$. We consider discretizations of integral operators $\mathscr{A}$ of the form

$$(\mathscr{A}v)(y) = \int_\Omega S(x,y)v(x)\,\mathrm{d}\mu_x,$$

where S is a positive singularity function. The test and ansatz functions ψ_i and φ_j are non-negative finite element functions satisfying $\operatorname{supp}\psi_i \subset D_1$ and $\operatorname{supp}\varphi_j \subset D_2$. The sub-block A results from a matrix partitioning which guarantees that the domains

$$D_1 = \bigotimes_{\nu=1}^{d}[a'_\nu, b'_\nu] \quad \text{and} \quad D_2 = \bigotimes_{\nu=1}^{d}[a_\nu, b_\nu]$$

are in the far-field of each other, i.e.,

$$\max_{\nu=1,\dots,d}\{b'_\nu - a'_\nu, b_\nu - a_\nu\} \le \eta\,\operatorname{dist}(D_1,D_2) \tag{3.84}$$

with a given parameter $\eta \in \mathbb{R}$.

The structure of this section is as follows. Before we show that $A_{1:m,j_{1:k}}$ and $A_{i_{1:k},1:n}$ can be approximated using Chebyshev polynomials, we present an alternative

formulation of ACA which allows to exploit the smoothness of the latter matrices for recompression. This second approximation requires the evaluation of the kernel function at transformed Chebyshev nodes, which has to be avoided if we want to use the original matrix entries. One solution to this problem is to find a least squares approximation. In Sect. 3.5.3 we show how this can be done in a purely algebraic and adaptive way. Numerical examples shows that this recompression is actually faster than the recompression technique from Sect. 2.6 and reduces the asymptotic storage complexity.

3.5.0.3 An Alternative Formulation of ACA

Since the methods of this section will be based on the pseudo-skeleton representation (3.81) of the ACA approximant, we should construct and store $A_{1:m,j_{1:k}}$, $A_{i_{1:k},1:n}$, and A_k^{-1} instead of UV^T. In order to generate and apply A_k^{-1} in an efficient way, we use the LU decomposition of A_k.

Assume that pairs (i_ℓ, j_ℓ), $\ell = 1,\dots,k$, have been found and assume that the normalized LU decomposition of the $k \times k$ matrix $A_k = L_k R_k$ has been computed. We find the new pivotal row i_{k+1} and column j_{k+1} as explained in Sect. 3.4. With the decomposition

$$A_{k+1} = \begin{bmatrix} A_k & b_k \\ a_k^T & c_k \end{bmatrix},$$

where $a_k := A_{i_{k+1},j_{1:k}}$, $b_k := A_{i_{1:k},j_{k+1}}$, and $c_k := A_{i_{k+1},j_{k+1}}$, the LU decomposition of A_{k+1} is given by

$$A_{k+1} = \begin{bmatrix} L_k & 0 \\ x_k^T & 1 \end{bmatrix} \begin{bmatrix} R_k & y_k \\ 0 & \alpha_k \end{bmatrix},$$

where x_k solves $R_k^T x_k = a_k$, y_k solves $L_k y_k = b_k$, and $\alpha_k = c_k - x_k^T y_k$. It is easy to see that

$$x_k^T = U_{i_{k+1},1:k}, \quad y_k^T = V_{j_{k+1},1:k}, \quad \text{and} \quad \alpha_k = (v_{k+1})_{j_{k+1}}.$$

This formulation of ACA has the same arithmetic complexity $\mathcal{O}(k^2(m+n))$ as the original formulation.

Combining ACA with the approximations (3.83) leads to

$$A \approx A_{1:m,j_{1:k}} A_{i_{1:k},j_{1:k}}^{-1} A_{i_{1:k},1:n} \approx B_1 C B_2^T,$$

where $C = X_1 A_k^{-1} X_2^T$. If $k' \le 2k$, then one should store the entries of $C \in \mathbb{R}^{k' \times k'}$. Otherwise, it is more efficient to store C in outer product form

$$C = (X_1 R_k^{-1})(X_2 L_k^{-T})^T,$$

which requires $2k'k$ units of storage.

3.5.1 Approximation Using Chebyshev Polynomials

Our aim in this section is to approximate the matrix $A_{1:m,j_{1:k}}$. The matrix $A_{i_{1:k},1:n}$ has a similar structure and its approximation can be done analogously. For notational convenience, we denote the restricted matrix $A_{1:m,j_{1:k}} \in \mathbb{R}^{m\times k}$ again by the symbol A. Hence, we assume that A has the entries (see (3.15))

$$a_{ij} = \Lambda_{1,i}\mathscr{A}\Lambda^*_{2,j}, \quad i = 1,\dots,m,\ j = 1,\dots,k. \tag{3.85}$$

Let $\mathfrak{I}_p = \mathfrak{I}_p^{(1)} \cdots \mathfrak{I}_p^{(d)}$ be the tensor product interpolation operator from (3.43) defined on Chebyshev nodes (3.39). According to Theorem 3.18 the interpolation on tensor product Chebyshev nodes leads to the following error estimate

$$|S(x,y) - \mathfrak{I}_{y,p}S(x,y)| \le \tilde{c}\, p \left(1 + \frac{2}{\pi}\log p\right)^d \left(\frac{\gamma\eta}{2+\gamma\eta}\right)^p |S(x,y)|. \tag{3.86}$$

We remind the reader of the representation (see (3.41)) of each one-dimensional interpolation operator

$$\mathfrak{I}^{(\nu)}_{y,p}S(x,y) = \sum_{i=0}^{p-1} c_i(x) T_i\left(2\frac{y^{(\nu)} - a_\nu}{b_\nu - a_\nu} - 1\right), \quad \nu = 1,\dots,d,$$

with coefficients

$$c_0(x) := \frac{1}{p}\sum_{j=0}^{p-1} S(x,t_j) \quad \text{and} \tag{3.87a}$$

$$c_i(x) := \frac{2}{p}\sum_{j=0}^{p-1} S(x,t_j)\cos i\frac{2j+1}{2p}\pi, \quad i = 1,\dots,p-1. \tag{3.87b}$$

The Chebyshev polynomials $T_p(t) := \cos(p\arccos(t))$ satisfy the three term recurrence relation

$$T_0(x) = 1, \quad T_1(t) = t, \quad \text{and} \quad T_{p+1}(t) = 2tT_p(t) - T_{p-1}(t), \quad p = 1,2,\dots. \tag{3.88}$$

Replacing the kernel function S with the polynomial $\mathfrak{I}_{y,p}S$, we define the operator

$$(\tilde{\mathscr{A}}v)(y) = \int_\Omega \mathfrak{I}_{y,p}S(x,y)v(x)\,\mathrm{d}\mu_x.$$

The corresponding matrix $\tilde{A}^{\mathrm{CH}} \in \mathbb{R}^{m\times k}$ has the entries

$$\tilde{a}^{\mathrm{CH}}_{ij} := \Lambda_{1,i}\tilde{\mathscr{A}}\Lambda^*_{2,j}, \quad i = 1,\dots,m,\ j = 1,\dots,k. \tag{3.89}$$

The following theorem shows the exponential convergence of $\tilde{A}^{\mathrm{CH}}$ for the case $c\gamma_1\eta < 1$.

Theorem 3.44. *If $c\gamma_1\eta < 1$ (with the constants from* (3.82)*), then the following error estimate is fulfilled*

$$\|A-\tilde{A}^{\mathrm{CH}}\|_F \le \bar{c}\,p\left(1+\frac{2}{\pi}\log p\right)^d\left(\frac{\gamma_1\eta}{2+\gamma_1\eta}\right)^p\|A\|_F.$$

Proof. With the positivity of S we obtain from (3.86) that

$$\begin{aligned}\|A-\tilde{A}^{\mathrm{CH}}\|_F^2 &= \sum_{i=1}^m\sum_{j=1}^k |a_{ij}-\tilde{a}_{ij}^{\mathrm{CH}}|^2\\ &= \sum_{i=1}^m\sum_{j=1}^k\left(\int_{\mathbb{R}^d}\int_{\mathbb{R}^d}|S(x,y)-\mathfrak{I}_{y,p}S(x,y)|\psi_i(y)\varphi_j(x)\,\mathrm{d}\mu_x\,\mathrm{d}\mu_y\right)^2\\ &\le \bar{c}^2p^2\left(1+\frac{2}{\pi}\log p\right)^{2d}\left(\frac{\gamma_1\eta}{2+\gamma_1\eta}\right)^{2p}\sum_{i=1}^m\sum_{j=1}^k|a_{ij}|^2\end{aligned}$$

and hence the assertion. The Nyström case and the collocation case follow from the same arguments. □

Instead of the singularity function S the kernel function of $\mathscr{A}$ may also contain normal derivatives of S. An example is the double-layer potential operator (see Sect. 3.1) arising in boundary element methods

$$(\mathscr{K}v)(y) = \frac{1}{4\pi}\int_\Gamma \frac{\nu_x\cdot(y-x)}{\|x-y\|^3}v(x)\,\mathrm{d}s_x,$$

where ν_x is the unit normal vector in $x\in\Gamma$. Since $\|x-y\|^{-3}$ is asymptotically smooth (with respect to both variables), we set for $i=1,\dots,m$ and $j=1,\dots,k$

$$\tilde{a}_{ij}^{\mathrm{CH}} := \int_\Gamma\int_\Gamma \nu_x\cdot(y-x)\mathfrak{I}_{y,p}\|x-y\|^{-3}\psi_i(y)\varphi_j(x)\,\mathrm{d}s_x\,\mathrm{d}s_y. \tag{3.90}$$

It is obvious that a similar error estimate as presented in Theorem 3.44 can be obtained also for this kind of operators.

3.5.2 Evaluation of the Approximation

The polynomial approximation (3.89) of the matrix entries a_{ij} has the form

$$\tilde{a}_{ij}^{\mathrm{CH}} = \sum_{\substack{\alpha\in\mathbb{N}_0^d\\ \|\alpha\|_\infty<p}} \underbrace{\Lambda_{1,i}\prod_{\nu=1}^d T_{\alpha_\nu}(\xi^{(\nu)})}_{=:b_{i\alpha}}\cdot\underbrace{\Lambda_{2,j}c_\alpha}_{=:X_{\alpha j}}, \qquad \xi^{(\nu)} = 2\frac{y^{(\nu)}-a_\nu}{b_\nu-a_\nu}-1,$$

with coefficient functions c_α. Hence, the matrix $\tilde{A}^{\mathrm{CH}}$ has the factorization $\tilde{A}^{\mathrm{CH}} = BX^{\mathrm{CH}}$, where $X^{\mathrm{CH}} \in \mathbb{R}^{k' \times k}$, $k' := p^d$, and $B \in \mathbb{R}^{m \times k'}$ has the entries

$$b_{i\alpha} = \Lambda_{1,i} \prod_{\nu=1}^{d} T_{\alpha_\nu}(\xi^{(\nu)}). \tag{3.91}$$

Let us first consider the collocation and the Nyström case. Then B has the entries

$$b_{i\alpha} = \prod_{\nu=1}^{d} T_{\alpha_\nu}(\xi_i^{(\nu)}), \quad \xi_i^{(\nu)} := 2\frac{y_i^{(\nu)} - a_\nu}{b_\nu - a_\nu} - 1.$$

This matrix can be computed efficiently using the recurrence relation (3.88). For all $\alpha \in \mathbb{N}^d$ satisfying $0 \le \alpha_\nu \le 1$, $\nu = 1,\ldots,d$, let $b_{i\alpha} = \prod_{\nu=1}^{d} (\xi_i^{(\nu)})^{\alpha_\nu}$. If there is $1 \le j \le d$ such that $\alpha_j \ge 2$, then

$$\begin{aligned} b_{i\alpha} &= T_{\alpha_j}(\xi_i^{(j)}) \prod_{\nu \ne j} T_{\alpha_\nu}(\xi_i^{(\nu)}) \\ &= 2\xi_i^{(j)} T_{\alpha_j - 1}(\xi_i^{(j)}) \prod_{\nu \ne j} T_{\alpha_\nu}(\xi_i^{(\nu)}) - T_{\alpha_j - 2}(\xi_i^{(j)}) \prod_{\nu \ne j} T_{\alpha_\nu}(\xi_i^{(\nu)}) \\ &= 2\xi_i^{(j)} b_{i\alpha - e_j} - b_{i\alpha - 2e_j}, \end{aligned}$$

where e_j is the jth canonical vector. Hence, the matrix B can be set up with $\mathcal{O}(mk')$ arithmetic operations.

Compute the vector $\tilde{v} = Xv$ and backup $\bar{v} = \tilde{v}$.
for $i = 1,\ldots,m$ **do**
  Restore $\tilde{v} = \bar{v}$.
  for $\alpha_2 = p, p-1, \ldots, 2$ **do**
    for $\alpha_1 = p, p-1, \ldots, 2$ **do**
      $\tilde{v}_{\alpha_1 - 1, \alpha_2} += 2\xi_i^{(1)} \tilde{v}_{\alpha_1, \alpha_2}$
      $\tilde{v}_{\alpha_1 - 2, \alpha_2} += -\tilde{v}_{\alpha_1, \alpha_2}$
    endfor
    $\tilde{v}_{0, \alpha_2} += \xi_i^{(1)} \tilde{v}_{1, \alpha_2}$
    $\tilde{v}_{0, \alpha_2 - 1} += 2\xi_i^{(2)} \tilde{v}_{0, \alpha_2}$
    $\tilde{v}_{0, \alpha_2 - 2} += -\tilde{v}_{0, \alpha_2}$
  endfor
  $w_i = \tilde{v}_{0,0} + \xi_i^{(2)} \tilde{v}_{0,1}$
endfor

Algorithm 3.5: Multiplication $w := BXv$

For computing the matrix-vector product $w := BXv$ without explicitly constructing B one can use the Clenshaw-like algorithm shown in Algorithm 3.5. This algorithm has complexity $\mathcal{O}(k'(k+m))$, and we state it for sake of simplicity for spatial dimension $d = 2$.

Now consider Galerkin matrices, i.e., ψ_i is a piecewise polynomial function on each of the M polyhedrons defining the computational domain. We assume that the support of each function ψ_i, $i = 1,\ldots,m$, is the union of at most μ elements τ_j,

$j \in \mathscr{I}_i$, with $|\mathscr{I}_i| \le \mu$, i.e., $\operatorname{supp} \psi_i = \bigcup_{j\in\mathscr{I}_i} \tau_j$. Each element τ_j is the image of the reference element $\hat{\tau}$ under a mapping F_j. The restriction of ψ_i to each polyhedron τ_j is a polynomial, and we apply a cubature formula

$$\int_{\hat{\tau}} f(y)\,\mathrm{d}\mu_y \approx \sum_{\ell=1}^{q} w_\ell f(y_\ell)$$

with weights w_ℓ and points y_ℓ, $\ell = 1,\dots,q$, on the reference element $\hat{\tau}$, i.e.,

$$b_{i\alpha} = \sum_{j\in\mathscr{I}_i} \int_{\tau_j} \prod_{\nu=1}^{d} T_{\alpha_\nu}(\xi^{(\nu)})\psi_i(y)\,\mathrm{d}\mu_y = \sum_{j\in\mathscr{I}_i}\sum_{\ell=1}^{q} w_\ell \psi_i(F_{\tau_j}(y_\ell)) \prod_{\nu=1}^{d} T_{\alpha_\nu}(F_{\tau_j}(\xi_\ell^{(\nu)})).$$

Let $\mathscr{I} := \bigcup_{i=1}^{m} \mathscr{I}_i$. The computation of the matrix B can therefore be done by first computing the matrix

$$b'_{(j,\ell),\alpha} := \prod_{\nu=1}^{d} T_{\alpha_\nu}(F_{\tau_j}(\xi_\ell^{(\nu)})), \quad \alpha \in \mathbb{N}^d,\ \|\alpha\|_\infty < p,\ j \in \mathscr{I},\ \ell = 1,\dots,q,$$

having at most μmq rows. The matrix B' has the same structure as B in (3.91) and the number of cubature nodes is bounded by $q = \mathscr{O}(k')$. In a second step one computes the matrix

$$c'_{i,(j,\ell)} := w_\ell \psi_i(F_{\tau_j}(y_\ell))$$

prior to computing the product

$$b_{i\alpha} = \sum_{j\in\mathscr{I}_i}\sum_{\ell=1}^{q} c'_{i,(j,\ell)} b'_{(j,\ell),\alpha}.$$

Note that the previous construction can be applied also to matrices (3.90).

As readily seen from (3.87), the computation of the coefficients X^{CH} requires additional evaluations of the kernel function at the tensor Chebyshev nodes. Since our aim is a method that is based on the matrix entries and does not require the kernel function, in the following we investigate a least squares approximation.

3.5.3 Least Squares Approximation

Let $B \in \mathbb{R}^{m\times k'}$ be the matrix defined in (3.91). According to Theorem 3.44, there is $X^{\mathrm{CH}} \in \mathbb{R}^{k'\times k}$ such that

$$\|A - BX^{\mathrm{CH}}\|_F \le \bar{c}\, p \left(1 + \frac{2}{\pi}\log p\right)^d \left(\frac{\gamma_1\eta}{2+\gamma_1\eta}\right)^p \|A\|_F.$$

We have pointed out that the computation of X^{CH} is not desirable. Additionally, there may be a matrix $X^{\mathrm{LS}} \in \mathbb{R}^{k' \times k}$ which provides a better approximation than X^{CH}. Hence, we aim at solving the least squares problem

$$\text{find } X \in \mathbb{R}^{k' \times k} \text{ such that } \|A - BX\|_F \text{ is minimized.}$$

Let $B = U_B \Sigma V_B^T$, $\Sigma \in \mathbb{R}^{k' \times k'}$, be a singular value decomposition of B, which can be computed with complexity $\mathcal{O}((k')^2 m)$. Then $X^{\mathrm{LS}} := V_B \Sigma^+ U_B^T A$, where

$$(\Sigma^+)_{ij} = \begin{cases} \sigma_i^{-1}, & i = j \text{ and } \sigma_i \neq 0, \\ 0, & \text{else,} \end{cases}$$

is a best approximation with respect to the Frobenius norm. The following error estimate for $\tilde{A}^{\mathrm{LS}} := BX^{\mathrm{LS}}$ is an obvious yet important consequence.

Lemma 3.45. *For the approximation $\tilde{A}^{\mathrm{LS}}$ we obtain*

$$\|A - \tilde{A}^{\mathrm{LS}}\|_F \leq \bar{c}\, p \left(1 + \frac{2}{\pi} \log p\right)^d \left(\frac{\gamma_1 \eta}{2 + \gamma_1 \eta}\right)^p \|A\|_F.$$

Proof. Since X^{LS} minimizes $\|A - BX\|_F$, we compare with the solution $\tilde{A}^{\mathrm{CH}} = BX^{\mathrm{CH}}$ defined by interpolation at the Chebyshev nodes. □

In what follows we will devise an efficient adaptive strategy for the solution of the least squares problem. According to the previous lemma, we may assume that

$$\|A - BX^{\mathrm{LS}}\|_F \leq \varepsilon \|A\|_F$$

with arbitrary $\varepsilon > 0$. Depending on, for instance, the geometry, the columns of B can be close to linearly dependent. Hence, the number of required columns of B may be significantly smaller than k'. Using the singular value decomposition of B, it is possible to construct a minimum orthonormal basis $U \in \mathbb{R}^{m \times k''}$ and coefficients $C \in \mathbb{R}^{k'' \times k'}$ such that

$$\|B - UC\|_F \leq \varepsilon \|B\|_F.$$

In this case we would have to store the matrix U for later computations. Since our aim is to generate the basis of approximation on the fly every time it appears in the computations, we have to find appropriate columns of B which are sufficient to represent the remaining columns. To this end, we construct a rank-revealing QR decomposition of B

$$B\Pi = QR = Q \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix},$$

where $Q \in \mathbb{R}^{m \times m}$ is unitary, $\Pi \in \mathbb{R}^{k' \times k'}$ is a permutation matrix, and $R \in \mathbb{R}^{m \times k'}$ is upper triangular. We determine $0 \leq r_B \leq k'$ such that $R_{11} \in \mathbb{R}^{r_B \times r_B}$ is non-singular and

$$\| \begin{bmatrix} 0 & R_{22} \end{bmatrix} X^{\mathrm{LS}} \|_F \leq \varepsilon \|A\|_F.$$

Denote by Π_{r_B} the first r_B columns of Π. Hence, setting $X_1 := [I, R_{11}^{-1}R_{12}]\Pi^{-1}X^{\mathrm{LS}}$, we have

$$\begin{aligned}
\|A - B\Pi_{r_B}X_1\|_F &\le \|A - BX^{\mathrm{LS}}\|_F + \|BX^{\mathrm{LS}} - B\Pi_{r_B}X_1\|_F \\
&\le \varepsilon\|A\|_F + \|B\Pi\Pi^{-1}X^{\mathrm{LS}} - B\Pi_{r_B}X_1\|_F \\
&= \varepsilon\|A\|_F + \| \left(\begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix} - \begin{bmatrix} R_{11} \\ 0 \end{bmatrix} \begin{bmatrix} I & R_{11}^{-1}R_{12} \end{bmatrix} \right) \Pi^{-1}X^{\mathrm{LS}}\|_F \\
&= \varepsilon\|A\|_F + \| \begin{bmatrix} 0 & R_{22} \end{bmatrix} \Pi^{-1}X^{\mathrm{LS}}\|_F \\
&\le 2\varepsilon\|A\|_F .
\end{aligned}$$

Although we have reduced the basis B to $B\Pi_{r_B}$, it still holds that

$$\min_{X \in \mathbb{R}^{r_B \times k}} \|A - B\Pi_{r_B}X\|_F \le 2\varepsilon\|A\|_F .$$

In addition to redundancies in the basis vectors B, the columns of A may be close to linearly dependent. An extreme case is $A = 0$. Then there is no need to store a coefficient matrix X of size $r_B \times k$. Therefore, our aim is to find $X \in \mathbb{R}^{r \times k}$ with minimum $0 \le r \le r_B$ such that

$$\|A - B\Pi_r X\|_F = \min_{Y \in \mathbb{R}^{r \times k}} \|A - B\Pi_r Y\|_F \le 2\varepsilon\|A\|_F .$$

Let $Q = [Q_1, Q_2]$ be partitioned, where $Q_1 \in \mathbb{R}^{m \times r}$. Since

$$\|A - B\Pi_r X\|_F = \|Q^T A - Q^T B\Pi_r X\|_F = \|Q^T A - \begin{bmatrix} \hat{R} \\ 0 \end{bmatrix} X\|_F = \| \begin{bmatrix} Q_1^T A - \hat{R}X \\ Q_2^T A \end{bmatrix} \|_F ,$$

where $\hat{R} \in \mathbb{R}^{r \times r}$ is the leading $r \times r$ submatrix in R_{11}, it follows that

$$\|A - B\Pi_r X\|_F = \|Q_2^T A\|_F$$

if X solves $\hat{R}X = Q_1^T A$. This $X \in \mathbb{R}^{r \times k}$ satisfies

$$\|A - B\Pi_r X\|_F = \min_{Y \in \mathbb{R}^{r \times k}} \|A - B\Pi_r Y\|_F .$$

The smallest r can thus be found from the condition

$$\|Q_2^T A\|_F \le 2\varepsilon\|A\|_F .$$

The computation of $Q^T A \in \mathbb{R}^{m \times k}$ can be done with $\mathcal{O}(kk'm)$ operations provided Q is represented by a product of k' Householder transforms.

In total, we have the following algorithm which requires $\mathcal{O}((k')^2 m)$ flops.

Set up the matrix $B \in \mathbb{R}^{m \times k'}$.
Compute an SVD $B = U_B \Sigma V_B^T$ and $X^{\mathrm{LS}} = V_B \Sigma^+ U_B^T A \in \mathbb{R}^{k' \times k}$.
Compute a rank-revealing QR decomposition $B\Pi = QR$.
Determine $0 \le r_B \le k'$ and partition $R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}$ such that $\| \begin{bmatrix} 0 & R_{22} \end{bmatrix} X^{\mathrm{LS}} \|_F \le \varepsilon \|A\|_F$.
Compute $Q^T A \in \mathbb{R}^{m \times k}$ by k' Householder transforms.
Find $0 \le r \le r_B$ such that $\|Q_2^T A\|_F \le 2\varepsilon \|A\|_F$.
Solve $\hat{R} X = Q_1^T A$ for $X \in \mathbb{R}^{r \times k}$.

Algorithm 3.6: Reduction of B and least squares solver.

Finally, in Table 3.13 we compare the asymptotic complexities of ACA, the recompression technique from this section (labeled "RACA"), and the standard method without any approximation.

Table 3.13 Asymptotic complexities.

	memory usage	matrix-vector multiplication	setup time
standard	mn	mn	mn
ACA	$k(m+n)$	$k(m+n)$	$k^2(m+n)$
RACA	kk'	$k'(m+n+k)$	$(k^2+(k')^2)(m+n)$

In what follows we discuss a further reduction in memory usage when the kernel is translationally invariant.

3.5.3.1 Translation Invariant Kernels

For simplicity we consider the Nyström case, i.e., our matrix block $A \in \mathbb{R}^{m \times k}$ is given by

$$a_{ij} = \kappa(x_j - y_i), \quad i = 1, \ldots, m,\ j = 1, \ldots, k.$$

In this case we simultaneously approximate all columns of A. The interpolation at tensor product Chebyshev nodes is given by

$$\tilde{a}_{ij} = (\mathfrak{I}_p \kappa)(x_j - y_i),$$

and allows for the error estimate in Theorem 3.44. The crucial point is the further reduction in storage, since we need only one vector of coefficients $\bar{X}^{\mathrm{CH}} \in \mathbb{R}^{k'}$, $k' = p^d$. In total, the matrix block $A \in \mathbb{R}^{m \times k}$ is compressed to k' coefficients – compared to kk' coefficients when no translation invariance is exploited.

3.5.4 Numerical Results

We consider the single-layer potential operator $\mathcal{V}$ from Sect. 3.1 for testing the proposed algorithm. In the following experiments Γ is the surface from Fig. 3.4.

A Galerkin discretization with piecewise constants $\varphi_i = \psi_i$, $i = 1,\dots,n$, leads to the matrix $V \in \mathbb{R}^{n\times n}$ with entries

$$v_{ij} = (\mathcal{V}\varphi_i, \varphi_j)_{L^2(\Gamma)}, \quad i,j = 1,\dots,n,$$

which is symmetric since $\mathcal{V}$ is self-adjoint with respect to $(\cdot,\cdot)_{L^2(\Gamma)}$. Therefore, it is sufficient to approximate the upper triangular part of V by an hierarchical matrix.

Tables 3.14 and 3.15 compare the hierarchical matrix approximation generated by ACA with and without coarsening (see Sect. 2.6) and by the (least squares based) method from this section, which is abbreviated with "RACA". We test these methods on three discretizations of the surface from Fig. 3.4. Columns two, five, and eight show the memory consumption in MByte, columns three, six, and nine contain the memory consumption per degree of freedom in KByte. The CPU time required for the construction of the respective approximation can be found in the remaining columns four, seven, and ten. The relative accuracy of the approximation in Frobenius norm is ε. All tests were done on a shared memory system with four Intel Xeon processors at 3 GHz. The minimal cluster size was chosen $n_{\min} = 15$, and $\eta = 1.1$.

Table 3.14 Approximation results for $\varepsilon = 1e-3$.

	ACA			coarsened ACA			RACA		
n	MB	KB/n	time	MB	KB/n	time	MB	KB/n	time
28 968	115.7	4.1	43s	93.0	3.3	46s	59.3	2.1	44s
120 932	607.8	5.1	230s	490.1	4.1	246s	244.3	2.1	237s
494 616	2836.6	5.9	1114s	2342.3	4.8	1175s	963.0	2.0	1155s

Table 3.15 Approximation results for $\varepsilon = 1e-4$.

	ACA			coarsened ACA			RACA		
n	MB	KB/n	time	MB	KB/n	time	MB	KB/n	time
28 968	186.2	6.6	63s	150.8	5.6	70s	127.4	4.5	66s
120 932	992.7	8.4	342s	809.8	6.9	372s	536.5	4.5	359s
494 616	4727.7	9.8	1674s	3928.4	8.1	1795s	2214.2	4.5	1714s

Apparently, RACA produces approximations with much lower memory consumption than obtained by the ACA method even after coarsening. The numbers in the column "KB/n" support our complexity estimates: The asymptotic complexity of the storage behaves linearly if the approximation accuracy is kept constant. The time required for RACA is less than the time for coarsened ACA. Especially the computation of the basis matrices can be done efficiently.

In the previous numerical results we have only computed $\mathcal{H}$-matrix approximations. Usually, also linear systems in which the approximant appears as the coefficient matrix have to be solved. Depending on the operator, the geometry of the

domain, and its discretization, preconditioning becomes necessary when these systems are to be solved iteratively. In the following chapter we present preconditioners which can be easily computed from the data-sparse approximation generated by ACA.

3.6 Preconditioning with Low-Accuracy Approximations

Building the coefficient matrices in boundary element applications is usually the most time-consuming part of the computation. By the ACA method presented in Sect. 3.4 it is possible to compute $\mathscr{H}$-matrices which approximate the original coefficient matrices up to any given accuracy with logarithmic-linear complexity. This construction can be accelerated on a parallel computer using the scheduling algorithms from Sect. 3.4.6 such that a complexity of order $p^{-1}n\log^{2(d-1)}n$ can be guaranteed for the construction of the linear system. In order to not destroy the overall complexity, also the solution of the linear system of equations should be computed with logarithmic-linear complexity.

Direct solvers will lead to a cubic asymptotic complexity due to the dense structure of the coefficient matrix. If the system is to be solved by an iterative scheme such as a Krylov subspace method, then the numerical range of the coefficient matrix will determine the required number of iterations. The condition number of Galerkin stiffness matrices arising from the discretization of pseudo-differential operators

$$\mathscr{A} : H^{\alpha}(\Gamma) \to H^{-\alpha}(\Gamma)$$

can be shown (see [182]) to be of the order $h^{-2|\alpha|}$, where $h := \min_{i=1,\ldots,n} \operatorname{diam} X_i$ is the discretization parameter. The coefficient matrices are therefore ill-conditioned if $\alpha \neq 0$ and if the number of unknowns n is large.

There are many different approaches to obtaining a preconditioner for matrices arising from the discretization of integral equations. One class of methods are additive or multiplicative Schwarz preconditioners [158, 177]. Another class of methods [257, 258, 47, 93, 173, 241, 172] is based on the ideas of multigrid, where instead of the usual application of multigrid methods, the FE discretization of differential operators, operators of negative order have to be treated. A third class [211, 212, 185] replaces the domain of integration of $\mathscr{A}$ in such a manner that the resulting matrix has Toeplitz structure. Another idea is based on the mapping properties of the operator only; see [244]. Let $\mathscr{A} : V \to V'$ and $\mathscr{B} : V' \to V$ be continuous, V-, resp., V'-coercive linear operators. Then $\mathscr{A}$ and $\mathscr{B}^{-1} : V \to V'$ both are V-coercive; i.e., there are constants $\alpha_{\mathscr{A}}, \alpha_{\mathscr{B}}, \beta_{\mathscr{A}}, \beta_{\mathscr{B}} > 0$ such that for all $v \in V$

$$\alpha_{\mathscr{A}}\|v\|_V^2 \leq (\mathscr{A}v, v)_{L^2} \leq \beta_{\mathscr{A}}\|v\|_V^2 \quad \text{and} \quad \alpha_{\mathscr{B}}\|v\|_V^2 \leq (\mathscr{B}^{-1}v, v)_{L^2} \leq \beta_{\mathscr{B}}\|v\|_V^2.$$

From the last estimates it already follows that $\mathscr{A}$ and $\mathscr{B}^{-1}$ are **spectrally equivalent**; i.e,

$$\frac{\alpha_{\mathscr{A}}}{\beta_{\mathscr{B}}}(\mathscr{B}^{-1}v,v)_{L^2} \leq (\mathscr{A}v,v)_{L^2} \leq \frac{\alpha_{\mathscr{B}}}{\beta_{\mathscr{A}}}(\mathscr{B}^{-1}v,v)_{L^2} \quad \text{for all } v \in V.$$

Hence, the hypersingular operator of the Laplacian can be used for preconditioning the single-layer potential operator, for instance. Since both operators can be approximated using ACA, one could easily obtain an efficient preconditioner. Instead of ACA, also the fast multipole method could be used. This, however, requires the implementation of the multipole method also for the preconditioning operator. Although the idea of using operators with inverse mapping properties is elegant, the preconditioning effect can be observed only asymptotically; i.e., for fixed problem sizes n the condition number might still be large. The double-layer potential operator, for instance, is asymptotically well-conditioned since it is an operator of order 0. We will however see from the numerical experiments in Sect. 3.6.2 that there are examples involving the double-layer operator which cannot be solved in reasonable time without preconditioning.

Since the inverse of an elliptic pseudo-differential operator is a pseudo-differential operator, too, it seems natural to use the (approximate) hierarchical inverse for preconditioning. Numerical experiments indicate that although the $\mathscr{H}$-matrix inverse provides a preconditioner with almost linear complexity, for dense matrices we are better off using

$$C := U_{\mathscr{H}}^{-1} L_{\mathscr{H}}^{-1}$$

as an explicit preconditioner, where $L_{\mathscr{H}}$ and $U_{\mathscr{H}}$ are lower and upper triangular $\mathscr{H}$-matrices such that $A_{\mathscr{H}} \approx L_{\mathscr{H}} U_{\mathscr{H}}$ is the approximate LU decomposition from Sect. 2.9. If $A_{\mathscr{H}}$ is Hermitian positive definite, $C := L_{\mathscr{H}}^{-T} L_{\mathscr{H}}^{-1}$ is used as a precon-

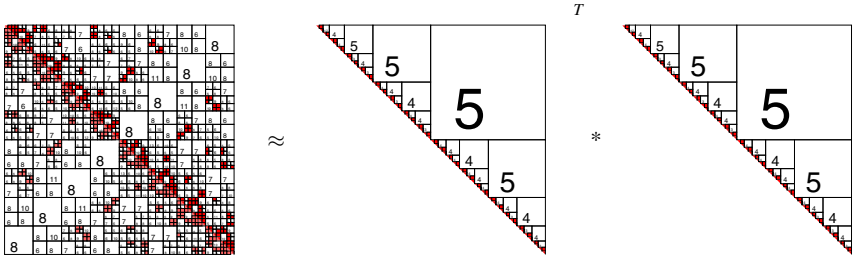


Fig. 3.8 Low-precision Cholesky decomposition.

ditioner, where $L_{\mathscr{H}}$ now is the lower triangular $\mathscr{H}$-matrix from the approximate Cholesky decomposition $A_{\mathscr{H}} \approx L_{\mathscr{H}} L_{\mathscr{H}}^T$. Hence, during an iterative scheme like GMRes, in addition to multiplications of $A_{\mathscr{H}}$ by a vector, forward/backward substitutions have to be applied when multiplying C by a vector. The forward/backward substitutions can be done with the same logarithmic-linear complexity as the $\mathscr{H}$-matrix-vector multiplication; see Sect. 2.9. In view of (2.23) and (2.26), the LU decomposition can be computed with lower precision δ compared with the accuracy ε of $A_{\mathscr{H}}$. For an improved efficiency, $A_{\mathscr{H}}$ should therefore be copied and treated

by the coarsening procedure from Sect. 2.6 such that we obtain $\tilde{A}_{\mathscr{H}} \in \mathscr{H}(T_{I\times I}, k)$ satisfying

$$\|A - \tilde{A}_{\mathscr{H}}\|_2 < \delta \|A\|_2.$$

We have seen in Sect. 2.9 that a hierarchical LU decomposition can be computed with complexity $k^2 n (\log n)^2$ such that

$$\|\tilde{A}_{\mathscr{H}} - L_{\mathscr{H}} U_{\mathscr{H}}\|_2 < \delta \|\tilde{A}_{\mathscr{H}}\|_2,$$

where for the following complexity analysis it is assumed that $k \sim (\log \delta)^{d-1}$. Note that this kind of dependence will be proved rigorously in the case of finite element discretizations of partial differential operators in the next chapter. From these estimates one obtains that

$$\begin{aligned}\|A - L_{\mathscr{H}} U_{\mathscr{H}}\|_2 &\leq \|A - \tilde{A}_{\mathscr{H}}\|_2 + \|\tilde{A}_{\mathscr{H}} - L_{\mathscr{H}} U_{\mathscr{H}}\|_2 < (1+\delta)\|A - \tilde{A}_{\mathscr{H}}\|_2 + \delta \|A\|_2 \\ &< \delta(2+\delta)\|A\|_2.\end{aligned}$$

According to Lemma 2.43, for a bounded condition number we have to impose the following condition on δ

$$\delta(2+\delta)\mathrm{cond}_2(A) \sim 1. \tag{3.92}$$

Following this condition, δ has to decrease if n increases, and we obtain an overall complexity $n(\log n)^6$ for the construction of the LU preconditioner in three spatial dimensions. From the numerical experiments, however, it will be seen that instead of condition (3.92) the weaker condition

$$\delta \sim 1$$

is sufficient. With this dependence, the complexity of the LU preconditioner can be estimated to be of the order $n(\log n)^2$. In either case, the hierarchical LU decomposition provides a spectrally equivalent preconditioner with almost linear complexity. It will be seen from the numerical experiments that the time for factorizing $A_{\mathscr{H}}$ can be neglected compared with building the approximant $A_{\mathscr{H}}$.

The ability to compute preconditioners from the matrix approximant $A_{\mathscr{H}}$ in a black-box way is one of the advantages of $\mathscr{H}$-matrices over fast multipole methods. The latter concentrate on the efficient multiplication of discrete operators by a vector, whereas the efficient representation of the whole discrete operator by $\mathscr{H}$-matrices allows to compute preconditioners from it. The usage of the LU decomposition as a fast direct solver will usually be less efficient than using it as a preconditioner since the approximate LU decomposition has to be computed with the (high) accuracy ε of $A_{\mathscr{H}}$.

The following numerical experiments demonstrate that the hierarchical LU decomposition can be efficiently used for preconditioning various kinds of problems resulting from industrial applications. We begin with the Dirichlet problem of the Laplacian on the complex geometry shown in Fig. 3.4. In the second set of numerical tests we consider regularized Neumann problems. The third part of our experiments treats mixed boundary value problems with almost vanishing Dirichlet part.

In order to be able to treat such problems with fast methods, we combine the stabilization technique from Sect. 2.5.1 with the hierarchical Cholesky decomposition. All test were performed on an AMD Athlon64 2.0 GHz workstation with 12 GB of core memory.

3.6.1 Dirichlet Problem

We return to the Dirichlet problem for the Laplacian from Sect. 3.4.5 on the geometry from Fig. 3.4. After the single-layer potential matrix $V_{\mathscr{H}}$ has been built in $\mathscr{H}$-matrix format, we recompress a copy of it to a prescribed accuracy $\delta \leq \varepsilon$ and compute the hierarchical Cholesky decomposition with precision δ. Depending on δ, this decomposition is either used as a preconditioner or for direct solution. In the latter case, we have to use $\delta = \varepsilon$ and may overwrite the matrix $V_{\mathscr{H}}$. Hence, no additional memory is needed in this case. If, however, the Cholesky decomposition is used for preconditioning, then δ is chosen 0.1.

In Table 3.16, the CPU time for recompressing a copy of $V_{\mathscr{H}}$ to accuracy δ, the memory consumption of the generated preconditioner and the CPU time for the Cholesky decomposition are presented. In these tests, η is chosen 1.0. Compared

Table 3.16 Computation of the hierarchical Cholesky decomposition.

	$n = 28\,968$			$n = 115\,872$		
δ	recompr.	MB	decomp.	recompr.	MB	decomp.
$1_{10}-1$	3.6s	11	3.4s	20.4s	54	25.6s
$1_{10}-2$	8.1s	40	5.7s	40.1s	224	53.0s
$1_{10}-3$	6.0s	73	21.4s	11.3s	366	135.1s
$1_{10}-4$		81	26.5s		410	173.0s

with building $V_{\mathscr{H}}$ (see Table 3.1), the CPU time for generating the Cholesky decomposition can be neglected if $\delta \sim 0.1$. The numerical effort increases significantly if $\delta = \varepsilon$. However, we still obtain a direct solver with relatively small effort.

In the final step of the computation we have to find the solution of the following approximate version of the linear system (3.70)

$$V_{\mathscr{H}} x = \tilde{b}, \qquad \tilde{b} = (\frac{1}{2} M + K_{\mathscr{H}})\tilde{g}. \tag{3.93}$$

Using the computed Cholesky decomposition of $V_{\mathscr{H}}$, we may find it directly as

$$x = L_{\mathscr{H}}^{-T} L_{\mathscr{H}}^{-1} \tilde{b},$$

if the precision δ of the decomposition is of the same order as the accuracy $\varepsilon = 1_{10}-4$ of $A_{\mathscr{H}}$. Table 3.17 shows the CPU time for solving (3.93) by hierarchical forward/backward substitution. The accuracy of the solution obtained by the direct

Table 3.17 Direct solution using hierarchical forward/backward substitution.

	time	$\|t-\tilde{t}_h\|_{L^2}$
28968	0.1s	$2.5_{10}-3$
115872	0.4s	$1.5_{10}-3$

method is of the same order as the iterative solution; cf. Table 3.4.

If on the other hand the Cholesky decomposition has been computed with low precision, we use it to solve (3.93) by the PCG. In Table 3.18 the number of iterations and the CPU time required to obtain a solution with residual norm $1_{10}-8$ are listed. As expected, the number of iterations is less the more accurate the *LU*

Table 3.18 Iterative solution using PCG.

	$n = 28968$		$n = 115872$	
δ	steps	time	steps	time
$1_{10}-1$	39	3.6s	40	20.1s
$1_{10}-2$	21	2.6s	21	14.1s
$1_{10}-3$	6	1.0s	6	5.2s

preconditioner is. In any case, the number of iterations does not depend on n. Although the iteration requires more time than the forward/backward substitution, the iterative solution is more efficient since computing the required Cholesky decomposition with high precision is much more expensive. As a conclusion, the number of iterations using the proposed preconditioner is bounded independently of n. A direct solution using backward/forward substitution is attractively fast. Factorizing A with high precision, however, is only worthwhile if (3.93) is to be solved for many right-hand sides $\tilde{b}$.

3.6.2 Neumann Problem

The second example is the boundary integral equation

$$\frac{1}{2}u(y)+\frac{1}{4\pi}\int_\Gamma u(x)\frac{\nu_x\cdot(y-x)}{\|x-y\|^3}\,\mathrm{d}s_x=\frac{1}{4\pi}\int_\Gamma\frac{\partial_{\nu_x}u(x)}{\|x-y\|}\,\mathrm{d}s_x,\quad y\in\Gamma,$$

with given Neumann boundary condition $\partial_\nu u = g$ on the surface Γ from Fig. 3.9, which is by courtesy of ABB Switzerland AG. This kind of integral equation arises from (3.13) in the case of pure Neumann boundary conditions. The dimension of the coefficient matrix A arising from a collocation method is $n = 3760$. We choose a small example to show that the presented methods are useful already for this problem size.

Since the kernel of A is one-dimensional (the surface is simply connected), the extended system

$$\begin{bmatrix} A & v \\ w^T & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix} \tag{3.94}$$

with $v \notin \mathrm{Im}A$ and $w \in \ker A$, which is uniquely solvable, has to be considered instead. Note that (3.94) arises from adding the auxiliary condition $x \perp \ker A$ by Lagrangian multipliers.

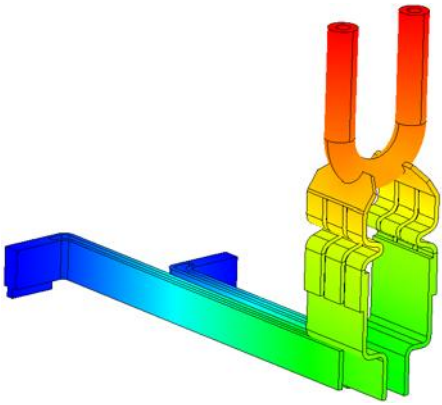


Fig. 3.9 Device with electric potential.

Theorem 3.46. *Let $A \in \mathbb{C}^{n\times n}$ and let the columns of $W \in \mathbb{C}^{n\times r}$ form a basis of the kernel $\ker A$ of A. Then the linear system*

$$\begin{bmatrix} A & V \\ W^H & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix} \tag{3.95}$$

with $V \in \mathbb{C}^{n\times r}$ is uniquely solvable provided that the columns of V form a basis of $\ker A^H$. For the solution of (3.95) it holds that $x \perp \ker A$. If $b \in \mathrm{Im}A$, then $y = 0$ and x is a solution of $Ax = b$. Otherwise, x is the pseudo-solution and Vy is its residuum.

Proof. Since $AW = 0$, we have

$$\begin{bmatrix} A & V \\ W^H & 0 \end{bmatrix} \begin{bmatrix} A & V \\ W^H & 0 \end{bmatrix}^H = \begin{bmatrix} AA^H + VV^H & 0 \\ 0 & W^HW \end{bmatrix}.$$

Hence, (3.95) is uniquely solvable if and only if

$$AA^H + VV^H = \begin{bmatrix} A & V \end{bmatrix} \begin{bmatrix} A & V \end{bmatrix}^H$$

is non-singular, which is satisfied if the columns of V are a basis of $(\mathrm{Im}A)^\perp = \ker A^H$. The augmented system (3.95) is equivalent with $Ax = b - Vy$ and $W^Hx = 0$.

Hence, x satisfies the normal equations $A^H Ax = A^H b$ due to $A^H V = 0$. Therefore, x is a solution of $Ax = b$ and $y = 0$ if and only if $b \in \mathrm{Im}A$. Otherwise, x is the pseudo-solution of $Ax = b$. □

If A is positive semi-definite, the augmented system (3.95) will be non-singular and symmetric, but it will not be positive definite. In fact, the augmented matrix from (3.95) is indefinite. This problem can be overcome by the following kind of regularization:

$$A' := A + VW^H,$$

where the columns of $V, W \in \mathbb{C}^{n\times r}$ form a basis of $\ker A^H$ and of $\ker A$, respectively.

Theorem 3.47. *The matrix A' is non-singular and the solution of $A'x = b$ satisfies $Ax = b$ provided that $b \in \mathrm{Im}A$. Otherwise x is a pseudo-solution of $Ax = b$. If A is positive semi-definite, then the matrix $A' := A + WW^H$ is positive definite.*

Proof. Let $x \in \mathbb{C}^n$. From $A^H V = 0$ it follows that

$$\|(A + VW^H)x\|_2^2 = \|Ax\|_2^2 + \|VW^H x\|_2^2 + 2\,\mathrm{Re}\, x^H A^H VW^H x = \|Ax\|_2^2 + \|VW^H x\|_2^2.$$

Therefore, $A'x$ can vanish only if $Ax = 0$ and $VW^H x = 0$. By the first condition we obtain that there is $z \in \mathbb{C}^r$ such that $x = Wz$. The second gives $VW^H Wz = 0$, which results in $z = 0$ due to the linear independence of the columns of V and W, respectively. Hence, $A'x = 0$ implies that $x = 0$. Since the solution of $A'x = b$ satisfies the normal equations $A^H Ax = A^H(A + VW^H)x = A^H b$, x is a solution of $Ax = b$ if this system is solvable. Otherwise x is the pseudo-solution of $Ax = b$.

It remains to show that $A + WW^H$ is positive definite. Since $\mathbb{C}^n = \ker A \oplus \mathrm{Im}A^H$, we use the decomposition $x = Wx_1 + Zx_2$ with $x_1 \in \mathbb{C}^r$, $x_2 \in \mathbb{C}^{n-r}$ and the columns of $Z \in \mathbb{C}^{n\times(n-r)}$ spanning $\mathrm{Im}A^H$. Since

$$x^H(A + WW^H)x = x^H Ax + \|W^H x\|_2^2 = x_2^H Z^H AZx_2 + \|W^H Wx_1\|_2^2,$$

we have $x^H A'x \geq 0$, and $x^H A'x = 0$ implies that $W^H Wx_1 = 0$ and $x_2^H Z^H AZx_2 = 0$. Since $W^H W$ is non-singular and $Z^H AZ$ is positive definite, it follows that $x_1 = 0 = x_2$. □

In Table 3.19 we compare the results obtained by fast methods and by a standard solution strategy; i.e., A is built without approximation and the augmented system (3.94) is solved using Gaussian elimination. For the kernel vectors we have used $v = w = (1,\ldots,1)^T$. In the row "Mbit", the results using an implementation of the fast multipole method can be found. The memory needed to store the matrix is reduced compared with the standard solution strategy. Although a fast method was used, it took longer to solve the system iteratively than solving it directly since no effective preconditioner was available. In the last row the results obtained by using ACA and the hierarchical LU preconditioner can be found. The preconditioner, which requires some additional storage, was computed from the approximant $A_{\mathscr{H}}$ of A with precision $\delta = 0.1$. The CPU time required for its computation, however, can be neglected. Using the $\mathscr{H}$-LU preconditioner, the iteration needs only few

Table 3.19 Solution of the regularized Neumann problem using different methods.

	storage		CPU time		
	matrix	precond.	matrix	precond.	solution
standard	108 MB		575.6s		1108.4s
Mbit	42 MB		149.1s		1273.2s
ACA	26 MB	12 MB	55.7s	1.7s	1.3s

steps to converge. Although the double-layer potential operator is asymptotically well-conditioned, the augmented coefficient matrix is ill-conditioned even for small problem sizes. This is due to the geometry and its discretization with strongly non-uniform triangles.

3.6.3 Mixed Boundary Value Problems

The problem we will consider in this section is the device from Fig. 3.10, which is connected to an electric source of opposite voltages on the dark parts of the left picture such that Dirichlet conditions are imposed on these subdomains. We compute the resulting current density j in the conductor with constant conductivity σ. Ampére's law for the electric field $\operatorname{curl} E = 0$ guarantees the existence of a scalar electric potential u satisfying $E = -\nabla u$. The continuity equation $\operatorname{div} j = 0$ together with Ohm's law $j = \sigma E$ leads to a boundary value problem of type (3.1) with $\mathscr{L} = -\Delta$ and Neumann conditions $\partial_\nu u = 0$ on the other part of the boundary. We apply ACA to the symmetric integral formulation (3.13); see [196], and [193]

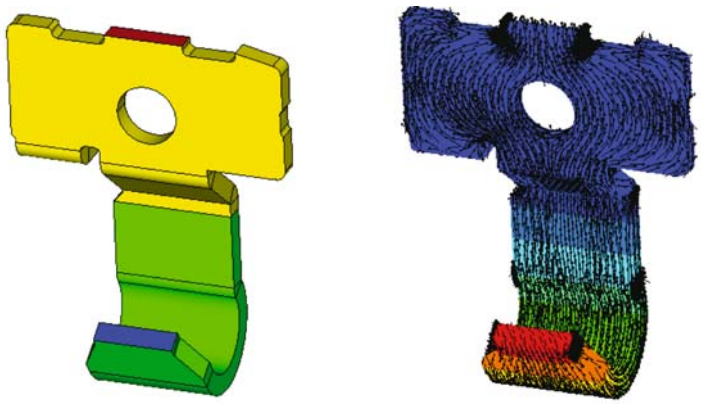


Fig. 3.10 Geometry with computed surface current.

for an application of the multipole method to mixed boundary value problems.

Due to (3.10) and (3.11), the discrete single-layer potential operator $V \in \mathbb{R}^{n_D \times n_D}$ and the discrete hypersingular operator $D \in \mathbb{R}^{n'_N \times n'_N}$ are symmetric positive definite matrices. Hence, the coefficient matrix

$$A := \begin{bmatrix} -V & K \\ K^T & D \end{bmatrix}$$

from (3.16) is symmetric and non-singular since the Schur complement $S := D + K^T V^{-1} K$ of $-V$ in A is symmetric positive definite. In fact (3.16) is a saddle-point problem; see [36]. For preconditioning A we could obviously approximate the following LU decomposition

$$A = \begin{bmatrix} L_1 & \\ -X^T & L'_2 \end{bmatrix} \begin{bmatrix} -L_1^T & X \\ & L_2'^T \end{bmatrix},$$

where $V = L_1 L_1^T$ is the Cholesky decomposition of V, X is defined by $L_1 X = K$, and $L'_2 L_2'^T = D + X^T X$ is the Cholesky decomposition of the Schur complement. The computation of $D + X^T X$, however, can be time-consuming even if $\mathscr{H}$-matrices are employed. Therefore, we use the following matrix C for symmetrically preconditioning A

$$C := \hat{U}^{-1} \begin{bmatrix} L_1^{-T} & \\ & L_2^{-T} \end{bmatrix}, \tag{3.96}$$

where

$$\hat{U} := \begin{bmatrix} I & -L_1^{-T} X \\ & I \end{bmatrix}$$

and L_1 and L_2 denote lower triangular $\mathscr{H}$-matrices such that

$$\|I - (L_1 L_1^T)^{-1} V\|_2 < \delta, \qquad \|I - (L_2 L_2^T)^{-1} D\|_2 < \delta, \tag{3.97}$$

and X is an $\mathscr{H}$-matrix satisfying $\|K - L_1 X\|_2 < \delta$. In order to guarantee that $L_1 L_1^T$ and $L_2 L_2^T$ are non-singular, we have to assume that $\delta < 1$; see Lemma 2.42. The size of δ which is sufficient to guarantee a bounded spectrum of $C^T A C$ will be estimated in Theorem 3.49. The computation of the Cholesky factors L_1 and L_2 with almost linear complexity was explained in Sect. 2.9. Note that L_2 is defined to be the approximate Cholesky factor of D but not of $D + X^T X$; i.e., instead of approximating the original coefficient matrix A, $C^T C$ approximates the matrix

$$\begin{bmatrix} -V & K \\ K^T & D - K^T V^{-1} K \end{bmatrix}^{-1}.$$

Our aim is to show that the spectrum of $C^T A C$ is enclosed in some small interval. From [240, p. 61 and Chap. 4] it is known that there are constants $\gamma_1, \gamma_2, \gamma_3 > 0$ such that

$$\gamma_1(Dx,x) \le (V^{-1}Mx,Mx) \le \gamma_2(Dx,x) \tag{3.98a}$$

$$(Dx,x) \le (Sx,x) \le \gamma_3(Dx,x) \tag{3.98b}$$

for all $x \in \mathbb{R}^{n'_N}$. Here, $M \in \mathbb{R}^{n_D \times n'_N}$ denotes the mass matrix.

Lemma 3.48. *The matrices $L_1L_1^T$, V and $L_2L_2^T$, S are spectrally equivalent, respectively; i.e.,*

$$(1-\delta)(L_1L_1^Tx,x) \le (Vx,x) \le (1+\delta)(L_1L_1^Tx,x) \quad \text{for all } x \in \mathbb{R}^{n_D}, \tag{3.99a}$$

$$\frac{1-\delta}{\gamma_3}(L_2L_2^Tx,x) \le (Sx,x) \le (1+\delta)(L_2L_2^Tx,x) \quad \text{for all } x \in \mathbb{R}^{n'_N}. \tag{3.99b}$$

Proof. For each eigenvalue λ of $(L_1L_1^T)^{-1}V$ with associated eigenvector $x \in \mathbb{R}^{n_D}$ satisfying $\|x\|_2 = 1$ we have

$$|1-\lambda| = \|x-(L_1L_1^T)^{-1}Vx\|_2 \le \|I-(L_1L_1^T)^{-1}V\|_2 < \delta$$

due to (3.97). Since $(L_1L_1^T)^{-1}V$ and $L_1^{-1}VL_1^{-T}$ are similar, one obtains from the variational representation of the smallest and the largest eigenvalue of $(L_1L_1^T)^{-1}V$ that

$$\lambda_{\min} = \inf_{x\in\mathbb{R}^n} \frac{(L_1^{-1}VL_1^{-T}x,x)}{(x,x)} = \inf_{x\in\mathbb{R}^n} \frac{(VL_1^{-T}x,L_1^{-T}x)}{(x,x)} = \inf_{y\in\mathbb{R}^n} \frac{(Vy,y)}{(L_1L_1^Ty,y)}$$

and

$$\lambda_{\max} = \sup_{x\in\mathbb{R}^n} \frac{(Vx,x)}{(L_1L_1^Tx,x)}.$$

Hence, it follows that

$$(1-\delta)(L_1L_1^Tx,x) \le (Vx,x) \le (1+\delta)(L_1L_1^Tx,x)$$

for all $x \in \mathbb{R}^{n_D}$. The same arguments together with (3.98b) prove the spectral equivalence of $L_2L_2^T$ and S. □

Theorem 3.49. *Let $0 < \delta < 1$. The eigenvalues of C^TAC are contained in*

$$[-1-c\delta,-1+c\delta] \cup [\gamma_3^{-1}(1-c\delta),1+c\delta],$$

where $c := 4(c_{\mathscr{K}}+2)\max\{\|V^{-1}\|_2,\|D^{-1}\|_2\}\max\{1,\delta(c_{\mathscr{K}}+2)\|V^{-1}\|_2\}+1$ and $c_{\mathscr{K}}$ is defined in (3.12).

Proof. We make use of the following symmetric decomposition of A:

$$A = U^T \begin{bmatrix} -V & \\ & S \end{bmatrix} U \quad \text{with} \quad U := \begin{bmatrix} I & -V^{-1}K \\ & I \end{bmatrix}.$$

Since

$$U\hat{U}^{-1} = \begin{bmatrix} I & -V^{-1}K \\ & I \end{bmatrix} \begin{bmatrix} I & L_1^{-T}X \\ & I \end{bmatrix} = \begin{bmatrix} I & -V^{-1}F \\ & I \end{bmatrix},$$

where $F := K - VL_1^{-T}X$, it follows that

$$C^T AC = \begin{bmatrix} -L_1^{-1}VL_1^{-T} & -L_1^{-1}FL_2^{-T} \\ -L_2^{-1}F^T L_1^{-T} & L_2^{-1}(S+F^T V^{-1}F)L_2^{-T} \end{bmatrix}.$$

We decompose $C^T AC$ into $C^T AC = B + E$, where

$$B := \begin{bmatrix} -L_1^{-1}VL_1^{-T} & \\ & L_2^{-1}SL_2^{-T} \end{bmatrix} \quad \text{and} \quad E := \begin{bmatrix} & -L_1^{-1}FL_2^{-T} \\ -L_2^{-1}F^T L_1^{-T} & L_2^{-1}F^T V^{-1}FL_2^{-T} \end{bmatrix}.$$

For the eigenvalues λ_j of $C^T AC$ it holds that $|\lambda_j(C^T AC) - \lambda_j(B)| \le \|E\|_2$, $j = 1,\dots,n$. Since

$$E = \begin{bmatrix} L_1^{-1} & \\ & L_2^{-1} \end{bmatrix} \begin{bmatrix} & -F \\ -F^T & F^T V^{-1}F \end{bmatrix} \begin{bmatrix} L_1^{-T} & \\ & L_2^{-T} \end{bmatrix}$$

has the same eigenvalues as

$$\begin{bmatrix} L_1 L_1^T & \\ & L_2 L_2^T \end{bmatrix}^{-1} \begin{bmatrix} & -F \\ -F^T & F^T V^{-1}F \end{bmatrix},$$

we obtain that

$$\|E\|_2 \le 2\|F\|_2 \max\{\|(L_1L_1^T)^{-1}\|_2, \|(L_2L_2^T)^{-1}\|_2\} \max\{1, \|V^{-1}\|_2\|F\|_2\}$$

due to Lemma 2.14. Since

$$F = [I - V(L_1L_1^T)^{-1}]L_1X + K - L_1X,$$

its norm is bounded by

$$\begin{aligned} \|F\|_2 &\le \|I - V(L_1L_1^T)^{-1}\|_2\|L_1X\|_2 + \|K - L_1X\|_2 \\ &\le \delta(\|K - L_1X\|_2 + \|K\|_2) + \|K - L_1X\|_2 \\ &\le \delta(1 + \delta + \|K\|_2) \le \delta(\|K\|_2 + 2). \end{aligned}$$

Additionally, $\|(L_1L_1^T)^{-1}\|_2 \le 2\|V^{-1}\|_2$ and $\|(L_2L_2^T)^{-1}\|_2 \le 2\|D^{-1}\|_2$ due to (3.97). Therefore, we have $\|E\|_2 \le c\delta$, where

$$c := 4(\|K\|_2 + 2)\max\{\|V^{-1}\|_2, \|D^{-1}\|_2\}\max\{1, \delta(\|K\|_2 + 2)\|V^{-1}\|_2\}.$$

The eigenvalues of B are the same as the eigenvalues of

$$\begin{bmatrix} -(L_1L_1^T)^{-1}V & \\ & (L_2L_2^T)^{-1}S \end{bmatrix}.$$

From Lemma 3.48 it follows that the eigenvalues of B are contained in

$$[-1-\delta,-1+\delta]\cup[\gamma_3^{-1}(1-\delta),1+\delta].$$

□

Since C^TAC is symmetric indefinite, we employ **MinRes** [199] for the iterative solution of the preconditioned linear system. Let r_k be the kth residual vector. Then for the convergence of MinRes it holds that

$$\|r_k\| \le 2\left(\frac{\sqrt{(a-\rho)(b+\rho)}-\sqrt{(a+\rho)(b-\rho)}}{\sqrt{(a-\rho)(b+\rho)}+\sqrt{(a+\rho)(b-\rho)}}\right)^{k/2}\|r_0\|, \quad k=1,2,\ldots,$$

where the spectrum is assume to be enclosed in positive and negative intervals $[-a-\rho,-a+\rho]\cup[b-\rho,b+\rho]$; see [117]. In order to obtain a convergence rate which is independent of the discretization, we therefore have to guarantee that $c\delta<\frac{1}{2}$. The latter condition means that δ has to decrease if n increases because c contains $\|V^{-1}\|_2$ and $\|D^{-1}\|_2$. The results of the preconditioned iterative solution are presented in Table 3.20. Each row contains the time required to compute the preconditioner C from (3.96), its memory consumption, the number of iterations for a relative error of $1_{10}-8$, and the time needed to compute the solution.

Table 3.20 Preconditioned MinRes for the domain from Fig. 3.10.

	$n=5\,154$					$n=20\,735$			
	precond.		solution			precond.		solution	
δ	time	MB	#It	time	δ	time	MB	#It	time
$5_{10}-2$	1.9s	6.7	11	0.5s	$5_{10}-3$	22.5s	35.6	9	3.0s
$1_{10}-1$	1.5s	5.6	17	0.6s	$1_{10}-2$	17.7s	34.7	13	4.1s
$2_{10}-1$	1.1s	4.5	23	0.8s	$5_{10}-2$	13.3s	30.5	24	5.9s

3.6.3.1 Mixed BVPs with Small Dirichlet Part

As the last example in this chapter we consider mixed boundary value problems with almost vanishing Dirichlet part; see the coil-like domain from Fig. 3.11. On the ends of the conductor we apply constant voltages of ± 1 V. Hence, this problem differs from the previous problem only by the computational geometry. The difficulty of this geometry is that the Dirichlet boundary, which somehow stabilizes the problem, is small compared with the Neumann part of the boundary. The coarsest discretization contains only 4 Dirichlet triangles. If the Dirichlet part would vanish completely, then the hypersingular operator would not be coercive at all. As a consequence, one can expect that a small Dirichlet boundary leads to a small coercivity constant in (3.11). Note that stabilizing this problem by the method applied

to the Neumann problem in Sect. 3.6.2 is physically wrong, since the matrix does not possess a kernel. The solution of this problem is a linearly increasing potential as shown in Fig. 3.11.

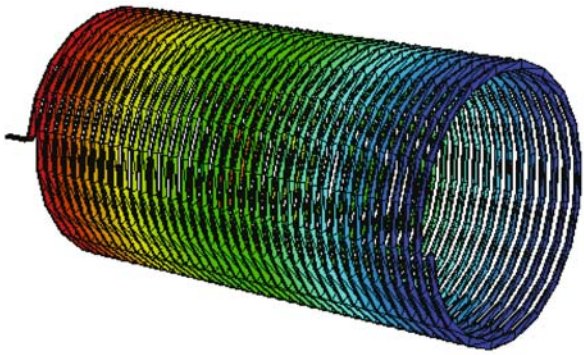


Fig. 3.11 Computational geometry with solution.

After generating the approximant with accuracy $\varepsilon = 1_{10}{-}6$, we recompress a copy of the coefficient matrix to a blockwise relative accuracy δ using the coarsening procedure from Sect. 2.6. The hierarchical Cholesky decomposition fails to compute unless we use a stabilized variant which is based on the stabilization technique from Sect. 2.5.1. The additional time for computing the preconditioner and

Table 3.21 Preconditioned MinRes for the domain from Fig. 3.11.

	$n = 3\,128$					$n = 12\,520$				
	precond.		solution			precond.		solution	
δ	time	MB	#It	time	δ	time	MB	#It	time
$1_{10}{-}3$	2.1s	9.3	55	1.4s	$2_{10}{-}4$	18.5s	36.0	50	4.8s
$2_{10}{-}3$	1.8s	8.5	69	1.7s	$5_{10}{-}4$	16.4s	32.8	71	7.2s
$5_{10}{-}3$	1.5s	7.5	84	2.1s	$1_{10}{-}3$	14.1s	30.1	96	8.9s

the time for the iterative solution in Table 3.21 can be neglected compared with the construction of the matrix approximant. Iterating without any preconditioner does not converge at all. Note that an asymptotic boundedness of the condition number with respect to n will not be enough in order to obtain a reasonable convergence behavior. The bad condition number is caused by the geometry and not by n.

Chapter 4
Application to Finite Element Discretizations

In this chapter we will apply the structure of $\mathscr{H}$-matrices to the finite element discretization of elliptic boundary value problems. We confine ourselves to Dirichlet problems

$$\mathscr{L}u = f \quad \text{in } \Omega, \tag{4.1a}$$
$$u = g \quad \text{on } \partial\Omega \tag{4.1b}$$

with bounded Lipschitz domains $\Omega \subset \mathbb{R}^d$, where $\mathscr{L}$ is a general uniformly elliptic second order partial differential operator in divergence form

$$\mathscr{L}u = -\operatorname{div}[C\nabla u + \gamma u] + \beta \cdot \nabla u + \delta u \tag{4.2}$$

with coefficients $c_{ij}, \beta_i, \gamma_i, \delta \in L^\infty(\Omega)$, $i, j = 1, \ldots, d$. The ellipticity of $\mathscr{L}$ is expressed by the assumption that for almost all $x \in \Omega$

$$0 < \lambda_{\mathscr{L}} \le \lambda(x) \le \Lambda_{\mathscr{L}} \tag{4.3}$$

for all eigenvalues $\lambda(x)$ of the symmetric matrix $C(x) \in \mathbb{R}^{d\times d}$ with entries c_{ij}. Additionally, we assume that there is a constant $\nu_{\mathscr{L}} \ge 0$ such that for almost all $x \in \Omega$

$$\lambda_{\mathscr{L}}^{-1}|\delta(x)| + \lambda_{\mathscr{L}}^{-2}\sum_{i=1}^{d} |\gamma_i(x)|^2 + |\beta_i(x)|^2 \le \nu_{\mathscr{L}}. \tag{4.4}$$

Note that we do not assume any smoothness of the coefficients. Even discontinuous coefficients are admitted. In this case, (4.1) is to be understood in the sense of a variational formulation

$$\text{find } u \in V \text{ such that } a(u,v) = l(v) \text{ for all } v \in V, \tag{4.5}$$

where the Sobolev space $H_0^1(\Omega)$ is used for V. The bilinear form a reads

M. Bebendorf, *Hierarchical Matrices*. Lecture Notes in Computational Science and Engineering 63,

$$a(u,v) = \int_\Omega \sum_{i,j=1}^{d} c_{ij}\partial_j u\,\partial_i v + \sum_{i=1}^{d} \beta_i v\,\partial_i u + \gamma_i u\,\partial_i v + \delta uv\,\mathrm{d}x \tag{4.6a}$$

$$= \int_\Omega \nabla v \cdot C\nabla u + v\beta\cdot\nabla u + u\gamma\cdot\nabla v + \delta uv\,\mathrm{d}x \tag{4.6b}$$

and the functional l takes the form

$$l(v) = \int_\Omega (f - \mathscr{L}\hat{g})v\,\mathrm{d}x,$$

where $\hat{g} \in H^1(\Omega)$ denotes the extension of the Dirichlet data $g \in H^{1/2}(\partial\Omega)$ to Ω. If $f \in H^{-1}(\Omega)$, $g \in H^{1/2}(\partial\Omega)$ and if $\mathscr{L}$ satisfies condition (4.3), (4.4) together with

$$\int_\Omega \delta v - \gamma\cdot\nabla v\,\mathrm{d}x \geq 0 \quad \text{for all } v \geq 0,\ v \in C_0^1(\Omega),$$

then (4.5) and (4.1) are uniquely solvable (see [104]) such that $\mathscr{L} : H_0^1(\Omega) \to H^{-1}(\Omega)$ is invertible.

In contrast to boundary element methods, the finite element method is based on the variational formulation of the boundary value problem (4.1) over Ω. For the numerical solution of the variational problem, Ω is usually replaced by a polyhedrization. For simplicity it is therefore assumed that Ω is polyhedral. Furthermore, in the conforming finite element method, $V = H_0^1(\Omega)$ is approximated by a space of piecewise linear ansatz functions $V_h \subset V$ with the discretization parameter $h := \max_{i\in I} \operatorname{diam} X_i$; i.e., for all $v \in V$

$$\inf_{v_h \in V_h} \|v - v_h\|_{H^1(\Omega)} \to 0 \quad \text{for } h \to 0.$$

From this discretization of (4.1) a linear system

$$Ax = b \tag{4.7}$$

arises, where together with A we define the following three $n \times n$ matrices,

$$A = \mathscr{J}^*\mathscr{L}\mathscr{J}, \quad B = \mathscr{J}^*\mathscr{L}^{-1}\mathscr{J}, \quad \text{and} \quad M = \mathscr{J}^*\mathscr{J}. \tag{4.8}$$

A is the stiffness matrix, B the Galerkin discretization of the inverse of $\mathscr{L}$, and M the mass matrix. Here, $\mathscr{J}$ is the natural bijection $\mathscr{J} : \mathbb{R}^n \to V_h$ defined by

$$\mathscr{J}x = \sum_{i\in I} x_i\varphi_i \tag{4.9}$$

and $\{\varphi_i\}_{i\in I}$ is the set of piecewise linear functions which form a basis of V_h, where $n := \dim V_h$ and $I = \{1,\ldots,n\}$ is used as an index set. Since $\mathscr{J}$ is also a function from $\mathbb{R}^n$ into V, the adjoint $\mathscr{J}^* \in L(V', \mathbb{R}^n)$ is defined by

$$(\mathscr{J}^*\varphi, x)_h = (\varphi, \mathscr{J}x)_{L^2} \quad \text{for all } x \in \mathbb{R}^n,\ \varphi \in V', \tag{4.10}$$

where $(x,y)_h := h^d \sum_{i\in I} x_i y_i$ denotes the (naturally scaled) Euclidean inner product.

The finite element matrices A and M are sparse and can therefore be stored in $\mathcal{O}(n)$ units of memory using sparse storage schemes such as CCS or CRS; see [222]. Since u is approximated by the finite element method in the volume Ω, the number of degrees of freedom n is typically significantly larger than in the boundary element method. Hence, the need for efficient numerical schemes is even higher. The **boundary concentrated finite element method** [157] is in some sense a mixture of the previous two methods.

Remark 4.1. Due to (4.3) and (4.4), the finite element stiffness matrix A is invertible for sufficiently small h. Since the principal part of $\mathscr{L}$ is coercive and the lower-order terms constitute a compact operator due to the compact embedding of L^2 in H^{-1}, $\mathscr{L}$ satisfies Gårding's inequality

$$(\mathscr{L}u,u)_{L^2(\Omega)} \geq \frac{\lambda_{\mathscr{L}}}{2}\|u\|^2_{H^1(\Omega)} - c\|u\|^2_{L^2(\Omega)} \quad \text{for all } u \in H^1(\Omega).$$

Due to the Céa-Polski Lemma (cf. [206]) A is invertible if h is sufficiently small.

Since the numerical solution of (4.7) is often the most time-consuming part of a computation, much attention has been paid to developing efficient solution strategies. There are two main classes of methods: *iterative* and *direct* solvers. The latter are based on factorizations of the sparse coefficient matrix A into easily invertible matrices. These methods are widely used due to their robustness. However, they suffer from so-called *fill-in*; i.e., compared with the sparsity of A considerably more entries of the factors will be nonzero; cf. [75]. Although this effect leads to super-linear complexity, constants are attractively small, making direct methods the methods of choice if n is not too large or if we are to solve problems in two spatial dimensions. Here, recent multifrontal solvers (see [78, 4, 230] and the references therein) can be used.

For higher spatial dimensions usually *iterative* methods such as Krylov subspace methods are more efficient, especially, if an approximate solution of relatively low accuracy is sought. The main advantage of these solution techniques is that the coefficient matrix enters the computation only through the matrix-vector product. On the other hand, the convergence rate and hence the number of iterations usually depends on the numerical range of A. Since $\mathscr{L}$ is a second order operator, the condition number of the FE stiffness matrix A in the symmetric case grows like $n^{2/d}$ for large n but also depends significantly on the coefficients of $\mathscr{L}$. By preconditioning one tries to improve the convergence properties of the Krylov subspace method. This usually requires tailoring the preconditioner C to the respective application.

For preconditioning, the best choice for C would obviously be $C = A^{-1}$. However, the computation of A^{-1} is usually more costly than solving $Ax = b$ for x. Hence, not A^{-1} but an approximation of A^{-1} is used for preconditioning. This idea gives rise to the class of approximate inverse preconditioners. The inverse of a FE stiffness matrix, however, is a dense matrix in general. In order to avoid large dense matrices, so-called **sparse approximate inverse** (SPAI) preconditioners (see [37, 123, 68, 67] and [38] for an overview) have been introduced. In this approach

the quantity $\|I - AC\|_F$ is minimized for matrices C having a given sparsity pattern. The minimum and hence the preconditioning effect, however, depend on the prescribed pattern. Additionally, if A is symmetric positive definite, the matrix C obtained from the minimization is not positive definite in general and cannot be used with the conjugate gradient method. To overcome this problem, the class of **factorized sparse approximate inverse** (FSAI) preconditioners [163, 162] was introduced; see also the AINV algorithm [37].

One of the best known and most often used preconditioning techniques is the **incomplete** LU **factorization** (ILU) and its variants; see [222]. Although the ILU can be applied to any sparse coefficient matrix provided that the factorization does not break down, it is well-suited for M- and diagonally dominant matrices. Similar to sparse inverses, the ILU is the LU decomposition computed on a given sparsity pattern, thereby avoiding fill-in. Generating an ILU decomposition is attractively fast. However, it usually does not lead to a bounded number of iterations of the solver. This goal can be achieved, for instance, by the **multigrid method** [125] or the **Bramble, Pasciak and Xu (BPX) preconditioner** [48] if the problem class is restricted to stiffness matrices of elliptic operators with smooth coefficients, where special properties of the operator can be exploited. For nonsmooth coefficients these methods might still work but suffer from poor convergence rates unless the respective method is adapted to the coefficients; see, for instance, [191]. In the past few years, much work has been done to develop robust multilevel methods. The **algebraic multigrid** (AMG) solvers [220] try to achieve this robustness by mostly heuristic strategies. A related class of preconditioners are the **AMLI preconditioners**; see [10, 11]. **Domain decomposition methods** [238] are commonly used if the computational domain can be subdivided into a small number of parts on each of which the coefficients do not vary too much. In this case, a problem can be decomposed into smaller ones for the purpose of parallel processing. One of the most widely used domain decomposition methods is the **FETI method** [89]. For a detailed survey on precondition techniques the reader is referred to [35].

The aim of this chapter is to lay theoretical ground to fast $\mathscr{H}$-matrix preconditioners for finite element systems. In Sect. 2.12 we have investigated the preconditioning effect of $\mathscr{H}$-matrix approximations. The efficiency of $\mathscr{H}$-matrices is not obtained from restricting the matrix to a given sparsity pattern, which is crucial for robustness. Therefore, we are enabled to approximate each entry of A^{-1} or the entries of the factors of the LU decomposition of A and hence capture more properties of A that are important for preconditioning. The preconditioner can, for instance, be guaranteed to be symmetric positive definite if A has this property; see Lemma 2.42. In [203] a similar approach for preconditioning was investigated. The inverse of the FE stiffness matrix is approximated on a matrix partition similar to the partition of a hierarchical matrix. But in contrast to $\mathscr{H}$-matrices, a blockwise constant approximation of the inverse was used, i.e., only one real number represents all entries of each block. Obviously, this needs less memory than storing a low-rank matrix and for preconditioning the approximant does not have to be too accurate. However, a certain accuracy is needed and the block constant approach is not adaptive, i.e., there is no possibility to guarantee a given accuracy on each block.

The mentioned preconditioning techniques usually guarantee a bounded condition number only with respect to n. The main advantage of $\mathscr{H}$-matrix preconditioners is that the number of iterations depends only on the accuracy of the approximation which can be arbitrarily chosen; see (2.24) and (2.27). This is of particular importance if a bad condition number is caused by the coefficients of the operator (see [112]) or by the computational domain. Due to approximation, the proposed class of preconditioners will be able to adapt itself to arbitrarily nonsmooth coefficients and, what is even more important for practice, does not require a grid hierarchy. The partition P on which the $\mathscr{H}$-matrix approximants will be constructed is the same as in the previous chapter (cf. (3.35) and Example 1.13) and can be generated using the following admissibility condition on the block $t \times s$, $t, s \subset I$,

$$\min\{\operatorname{diam} X_t, \operatorname{diam} X_s\} \leq \eta \operatorname{dist}(X_t, X_s) \tag{4.11}$$

for some parameter $\eta > 0$, where as usual we set

$$X_t := \bigcup_{i \in t} X_i \quad \text{for } t \subset I \quad \text{and} \quad X_i := \operatorname{supp} \varphi_i.$$

Note that the previous condition does not depend on the operator (4.2). In fact it will be shown that this condition is sufficient to prove existence of $\mathscr{H}$-matrix approximants of logarithmic-linear complexity for any elliptic operator. The structure of low-rank matrices on each sub-block will be able to account for the local properties of the operator. Interestingly, this will hold true for both the inverse and the factors of the LU decomposition. The robustness, however, comes with the disadvantage that $\mathscr{H}$-matrix preconditioners usually require more resources than a preconditioner that has been optimized for a special application. Nevertheless, it will be possible to generate, store, and apply the $\mathscr{H}$-matrix preconditioner to a vector with logarithmic-linear complexity.

For all results in this chapter that are based on the *geometric* admissibility condition (4.11) we will assume quasi-uniform and shape regular discretizations. We have pointed out in Sect. 1.4.1 that for a cluster tree which is geometrically and cardinality balanced it is usually required that the underlying grid satisfies the previous assumption. However, the solution of boundary value problems (4.1) usually contains certain types of singularities, which have to be accounted for by local grid refinement. The results that are based on the *algebraic* admissibility condition (1.13) will admit general discretizations.

For the computation of the inverse and of the LU decomposition in the algebra of $\mathscr{H}$-matrices it is of particular importance that the sparse finite element matrix A is an $\mathscr{H}$-matrix. If $b \in P$ satisfies (4.11), then the supports of the basis functions are pairwise disjoint. Hence, the matrix entries in this block vanish. According to Remark 2.7 it holds that $A \in \mathscr{H}(T_{I\times I}, n_{\min})$, which leads to logarithmic-linear storage costs. The following lemma shows that the storage requirements (and hence the number of operations of the matrix-vector multiplication) are actually linear.

Lemma 4.2. *Storing a FE matrix A as an $\mathscr{H}$-matrix requires $\mathcal{O}(n)$ units of storage.*

Proof. Since A vanishes on admissible blocks, we only have to estimate the number of non-admissible blocks. Let $t \in T_I^{(\ell)}$ be a cluster from the ℓth level of T_I. The number of elements of the set

$$N(t) := \left\{ s \in T_I^{(\ell)} : \eta \operatorname{dist}(X_t, X_s) \leq \min\{\operatorname{diam} X_t, \operatorname{diam} X_s\} \right\}$$

is bounded by a constant since due to (1.22)

$$\begin{aligned} |N(t)| 2^{-\ell} / c_G &\leq \sum_{s \in N(t)} \mu(X_s) \leq \nu \mu(X_{N(t)}) \\ &\leq \nu c_\Omega (\operatorname{diam} X_t + \eta^{-1} \min\{\operatorname{diam} X_t, \operatorname{diam} X_s\})^d \\ &= \nu c_\Omega (1 + 1/\eta)^d c_g 2^{-\ell} \end{aligned}$$

gives $|N(t)| \leq \nu c_g c_G c_\Omega (1 + 1/\eta)^d$. Since there are at most $|T_I| \sim |I|$ cluster $t \in T_I$, we obtain the desired result. □

Approximations of the inverse or of the LU decomposition can be used also for the direct solution of the linear system (4.7). Compared with the purpose of preconditioning, the approximation accuracy in this case has to be much higher. As a consequence, the usage in an iterative scheme will outperform the direct approach. The usage as a direct solver may be beneficial if (4.7) is to be solved for many right-hand sides b.

In order to be able to guarantee a robust and efficient preconditioner, it is indispensable to analyze the complexity of the proposed methods. Note that for the complexity analysis of the inversion and of the LU factorization algorithms from Sect. 2.8 and Sect. 2.9 we have assumed that the blockwise rank is bounded. In Sect. 2.12 we could not specify how the blockwise rank depends on the approximation accuracy. Although any matrix can be approximated by an $\mathscr{H}$-matrix, the required blockwise rank might be full. Since it is by no means obvious that reasonable approximants exist, we have to investigate the existence of $\mathscr{H}$-matrix approximants and the rank which is associated with a given accuracy.

The structure of this chapter is as follows. In Sect. 4.1 it will be proved that the inverse of A can be approximated by $\mathscr{H}$-matrices with logarithmic-linear complexity. The estimates show that the $\mathscr{H}$-matrix approximation is robust in the sense that the efficiency does depend neither on the shape of the domain Ω nor on the smoothness and only slightly on the size of the coefficients of $\mathscr{L}$.

Although an almost linear complexity can be observed, the $\mathscr{H}$-matrix inverse will be used only for theoretical purposes. The hierarchical LU decomposition from Sect. 2.9 will turn out to be significantly more efficient. In contrast to the inverse, the LU decomposition has no analytic analogue. Hence, the existence of $\mathscr{H}$-matrix approximants cannot rely on analytic properties. In Sect. 4.2 it will be proved that Schur complements of finite element stiffness matrices can be approximated, which will be used to show the existence of $\mathscr{H}$-matrix approximants to the factors of LU decompositions in Sect. 4.3. The numerical experiments from Sect. 4.4 will demonstrate the efficiency and the robustness of the $\mathscr{H}$-matrix LU decomposition. Nested

dissection reorderings improve the efficiency of the $\mathscr{H}$-matrix LU factorization because the problem is transfered to interfaces, which in particular allows to parallelize the algorithm; see Sect. 4.5. The size of the interfaces can be minimized if the matrix partition is constructed using the matrix graph instead of the geometrical information.

Furthermore, we use $\mathscr{H}$-matrices for the solution of nonlinear problems. In Sect. 4.6 we present a method which is related to Broyden's update formula. It explicitly updates the factors of an initially computed LU decomposition of a discrete Jacobian each time a Broyden update is applied to it, which avoids storing the history of update vectors. The approximation by hierarchical matrices introduces additional errors which can deteriorate the convergence of Broyden's method. We derive a condition under which the super-linear convergence of Broyden's method is preserved.

4.1 Approximating FE Matrix Inverses

In this section we investigate the existence of $\mathscr{H}$-matrix approximants to the inverse of finite element discretizations of boundary value problems (4.1). We have learned from Chap. 3 on integral operators that a local approximation of the kernel function by a degenerate function can be exploited to define low-rank approximants of the discrete operator; see (3.28). A similar technique will be used to prove that the discretization B of $\mathscr{L}^{-1}$ can be approximated by $\mathscr{H}$-matrices. For this purpose we will use the following relation between $\mathscr{L}^{-1}$ and the **Green's function** G for $\mathscr{L}$ and Ω:

$$(\mathscr{L}^{-1}\varphi)(y) = \int_\Omega G(x,y)\varphi(x)\,\mathrm{d}x \quad \text{for all } \varphi \in C_0^\infty(\Omega). \tag{4.12}$$

In Chap. 3 the kernel approximants were mostly generated by interpolation. Hence, we took advantage of the property that the kernel function of the operator is C^∞ apart from the diagonal $x = y$. This smoothness is due to the fact that integral methods are applied only to differential operators with constant or at least smooth coefficients. The main advantage of the finite element method, however, is that it can be applied with very weak assumptions on the coefficients. Measurable coefficients are usually sufficient. Hence, the Green's function will not be smooth such that the interpolation theory from Chap. 3 cannot be applied here. The lack of regularity will be overcome by a technique which is based on a Caccioppoli inequality; Sect. 4.1.4. As a result, it will be shown that G can be approximated on domains satisfying (4.11), which leads to the existence result for the discretization B of $\mathscr{L}^{-1}$. The blockwise rank will be shown to depend logarithmically on the approximation accuracy. Since we are not only interested in the discretization of $\mathscr{L}^{-1}$ but in the inverse of the finite element stiffness matrix A, which do not coincide in general, we have to exploit a relation between A^{-1}, B, and M^{-1} in Sect. 4.1.5. Note that a Caccioppoli inequality was already used as one of the key components in the proof of Lemma 3.5. The second component was estimate (3.23) which, however, does not hold for differen-

tial operators with arbitrary coefficients. The numerical results in Sect. 4.1.6 will show an almost linear complexity of the inversion procedure from Sect. 2.8 and hence confirm our analysis. It will be seen that $\mathscr{H}$-matrices are robust in the sense that their efficiency does not depend on the smoothness and only slightly on the size of the coefficients. In Sect. 4.1.3 we will present an algebraic approach to the approximation of the inverse of matrices which are sparse in the sense of (1.33). In particular, this approach allows to treat general grids. Furthermore, we prove that the so-called *weak admissibility condition* (cf. [134], which improves the a-priori block structure, leads to the same rank estimates as the admissibility condition (4.11)). From these results it will be seen that the complexity of the approximation depends at most logarithmically on the size of the coefficients.

Before we turn to the approximation of the Green's function, in Sect. 4.1.1 we first briefly review the literature on inverses of banded matrices and deduce properties that will be used for the existence of $\mathscr{H}$-matrix approximants to the inverse mass matrix M^{-1} in Sect. 4.1.2. Furthermore, we present a first algebraic approach to the approximation of inverses of general sparse matrices by $\mathscr{H}$-matrices in Sect. 4.1.3.

4.1.1 Inverses of Banded Matrices

A matrix A is said to be (p,q)**-banded** with some $p,q \in \mathbb{N}$ if $a_{ij} = 0$ for $p < j - i < -q$. A **strictly** (p,q)**-banded** is a (p,q)-banded matrix with non-vanishing elements in the pth diagonal above and the qth diagonal below the main diagonal. Since (p,q)-banded matrices contain many zero, the entries of its inverse have to depend on each other to some extend. However, it is not obvious which is the right structure to account for the contained redundancy. The following definition of a Green's matrix is due to Asplund [7].

Definition 4.3. A square matrix A is called **Green's matrix** of grade p if $\operatorname{rank} A_{ts} \leq p$ for all blocks $t \times s$ satisfying $p + \min s > \max t$. A is called Green's matrix of grade (p,q) if A^T is additionally a Green's matrix of grade q.

The following theorem (see [15]) relates the vanishing of entries in a banded matrix to the vanishing of a corresponding set of minors in the inverse.

Theorem 4.4. *A non-singular square matrix A is (p,q)-banded if and only if A^{-1} is a Green's matrix of grade (p,q).*

In [95] it was first observed that the inverse of a symmetric, non-singular matrix is an irreducible tridiagonal matrix if and only if A^{-1} is given by two vectors $u \in \mathbb{R}^n$ and $v \in \mathbb{R}^n$ (so-called generator vectors) such that

$$(A^{-1})_{ij} = \begin{cases} u_i v_j, & i \leq j, \\ v_i u_j, & j \leq i. \end{cases}$$

The inverse can hence be represented by two half rank-1 matrices, i.e., semi-separable matrices; cf. Definition 1.10. The following example shows that the presence of non-vanishing entries in the subdiagonals is important for the existence of generating vectors.

Example 4.5. The symmetric tridiagonal matrix

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

is its own inverse and cannot be represented by two vectors u and v.

For strictly banded matrices the following result was obtained by Rósza in [219]. Notice that the inverses satisfy a stronger condition than semi-separability.

Theorem 4.6. *A non-singular matrix is strictly (p,q)-banded if and only if its inverse has the representation*

$$(A^{-1})_{ij} = \begin{cases} (UV^H)_{ij}, & i < j+p, \\ (WZ^H)_{ij}, & j < i+q \end{cases}$$

for some matrices $U,V \in \mathbb{R}^{n\times p}$ and $W,Z \in \mathbb{R}^{n\times q}$ satisfying $(UV^H)_{ij} = (WZ^H)_{ij}$ if $-p < j-i < q$ and $(UV^H)_{ij} \neq (WZ^H)_{ij}$ if $j+p = i$ or $i+p = j$.

Example 4.7. The inverse of the strictly $(1,1)$-banded matrix

$$A = \begin{bmatrix} 2 & -1 & & & & & & \\ -1 & 2 & -1 & & & & & \\ & -1 & 2 & -1 & & & & \\ & & -1 & 2 & -1 & & & \\ & & & -1 & 2 & -1 & & \\ & & & & -1 & 2 & -1 & \\ & & & & & -1 & 2 & -1 \\ & & & & & & -1 & 2 \end{bmatrix},$$

which results from discretizing two-point boundary value problems

$$\begin{aligned} -u'' &= f \quad \text{in } \Omega := (a,b), \\ u(a) &= u(b) = 0 \end{aligned}$$

can be calculated to be

$$A^{-1} = \frac{1}{9}\begin{bmatrix} 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 7 & 14 & 12 & 10 & 8 & 6 & 4 & 2 \\ 6 & 12 & 18 & 15 & 12 & 9 & 6 & 3 \\ 5 & 10 & 15 & 20 & 16 & 12 & 8 & 4 \\ 4 & 8 & 12 & 16 & 20 & 15 & 10 & 5 \\ 3 & 6 & 9 & 12 & 15 & 18 & 12 & 6 \\ 2 & 4 & 6 & 8 & 10 & 12 & 14 & 7 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}.$$

With $u = \frac{1}{3}[8,7,6,5,4,3,2,1]^T$ and $v = \frac{1}{3}[1,2,3,4,5,6,7,8]^T$ we have that (see the notation from Sect. 1.2)

$$A^{-1} = \mathrm{triu}(uv^T, 0) + \mathrm{tril}(vu^T, 1).$$

The presence of the above structures, which allow an exact and efficient factorization, is however restricted to one-dimensional problems. The bandwidth of matrices resulting from the discretization of differential operators in two or more spatial dimensions usually depends on n even if the bandwidth has been reduced by reordering the indices. The minimal bandwidth in two dimensions is for instance of the order $\sqrt{n}$. The results from this section are more or less useless for such matrices since the rank of semi-separable matrices will be of the order $\sqrt{n}$, which leads to a complexity $n^{3/2}$ when storing the generator matrices. In addition, the decay rate in (4.13) will be much smaller. For higher dimensions, fast algorithms require approximation; see for instance [151] in which the inverse is generated in compressed form.

For the inverse of banded matrices an exponential decay of the entries a_{ij} with respect to the distance $|i-j|$ has been observed; cf. [77, 189].

Theorem 4.8. *Let A be a symmetric positive definite (p,p)-banded matrix. Then*

$$|(A^{-1})_{ij}| \le 2\lambda^{2|i-j|/p}\|A^{-1}\|_2, \tag{4.13}$$

where $\lambda = (\sqrt{\mathrm{cond}_2(A)} - 1)/(\sqrt{\mathrm{cond}_2(A)} + 1)$.

Theorem 4.8 can be generalized for later purposes by considering the minimal length d_{ij} of paths connecting i and j within the matrix graph $G(A)$ (see (1.11)) instead of $|i-j|$. In Example 1.14 we have assumed that A is irreducible. If A is reducible, then there is $(i,j) \in I \times I$ for which no path from i to j exists. In this case, we formally set $d_{ij} = \infty$ because $(A^{-1})_{ij} = 0$.

Notice that the distance d_{ij} of two vertices $i, j \in I$ in the matrix graph $G(A)$ is a metric. To see this, it is sufficient to show the triangle inequality $d_{ij} \le d_{i\ell} + d_{\ell j}$ for all $\ell \in I$. Assume there exists $\ell \in I$ such that $d_{i\ell} + d_{\ell j} < d_{ij}$. Then there is a path from i to j over ℓ that is shorter than a minimal path, which is a contradiction to the definition of d_{ij}.

The following lemma is a consequence of $a_{ij} = 0$ for all (i,j) satisfying $d_{ij} > 1$.

Lemma 4.9. *Let $k \in \mathbb{N}$. Then $(A^k)_{ij} = 0$ for all $i, j \in I$ satisfying $d_{ij} > k$. Furthermore, $[(A^H A)^k]_{ij} = 0$ for all $i, j \in I$ satisfying $d_{ij} > 2k$.*

Proof. The result is obvious for $k = 0$ and $k = 1$ due to the definition of d_{ij}. Let $B = A^{k-1}$. Then by induction it holds that $b_{ij} = 0$ for pairs (i, j) satisfying $d_{ij} > k-1$. From $k < d_{ij} \le d_{i\ell} + d_{\ell j}$ for all $\ell \in I$ it follows that $d_{i\ell} > k-1$ or $d_{\ell j} > 1$. Hence, $b_{i\ell} = 0$ or $a_{\ell j} = 0$, which gives

$$(A^k)_{ij} = \sum_{\ell \in I} b_{i\ell} a_{\ell j} = 0.$$

The second part of the assertion follows from similar arguments. □

Theorem 4.10. *Let A be a symmetric positive definite matrix. Then*

$$|(A^{-1})_{ij}| \le 2\lambda^{d_{ij}} \|A^{-1}\|_2$$

with λ defined in Theorem 4.8.

Proof. We may assume that $d_{ij} > 0$ since $|(A^{-1})_{ij}| \le \|A^{-1}\|_2$. For any polynomial $p \in \Pi_k$ with $k < d_{ij}$ it follows from Lemma 4.9 that $[p(A)]_{ij} = 0$. Furthermore, the spectral norm and the spectral radius coincide for normal matrices:

$$\|A^{-1} - p(A)\|_2 = \rho(A^{-1} - p(A)) = \max_{x \in \sigma(A)} |x^{-1} - p(x)|.$$

A result due to Chebyshev says that Π_k contains a polynomial p_k (cf. [183, p. 33]) so that

$$\|x^{-1} - p_k(x)\|_{\infty,[a,b]} \le c\,\lambda^{k+1}$$

with

$$c = \frac{(1+\sqrt{r})^2}{2ar} \le \frac{2}{a}, \quad \lambda = \frac{\sqrt{r}-1}{\sqrt{r}+1}, \quad r = \frac{b}{a}.$$

Let $a = \|A^{-1}\|_2^{-1}$ and $b = \|A\|_2$. Then $\sigma(A) \subset [a, b]$ and $a > 0$ since A is positive definite. Setting $k = d_{ij} - 1$, the previous arguments prove

$$|(A^{-1})_{ij}| = |(A^{-1})_{ij} - [p_k(A)]_{ij}| \le \|A^{-1} - p_k(A)\|_2 \le c\,\lambda^{k+1} = c\,\lambda^{d_{ij}},$$

which gives the assertion. □

The previous theorem can be generalized to invertible matrices A by applying it to $A^T A$. The respective estimate can be obtained from

$$|(A^{-1})_{ij}| = |((A^T A)^{-1} A^T)_{ij}| \le \|A\|_\infty \max_{i,j} |((A^T A)^{-1})_{ij}|.$$

Although the previous result is a generalization to multi-dimensional problems, its application to FE stiffness matrices is not helpful since the condition number of FE stiffness matrices depends on n, which destroys the exponential decay. Notice that the entries of A^{-1} from Example 4.7 do decay but they do not decay exponentially. Theorem 4.10, however, can be used for proving that the inverse mass matrix can be approximated, because the FE mass matrix is well-conditioned.

4.1.2 Approximating the Inverse Mass Matrix

The inverse of the mass matrix M will arise when the inverse stiffness matrix A^{-1} will be related with the discrete inverse B. Therefore, $\mathscr{H}$-matrix properties of M^{-1} are to be investigated. In order to avoid technical complications, quasi-uniform and shape-regular triangulation (see (1.17) and (1.18)) are considered, i.e.,

$$\operatorname{vol} X_i \geq c_v h^d. \tag{4.14}$$

The supports X_i may overlap. In accordance with the standard finite element discretization we require that each triangle belongs to the support of a bounded number of basis functions; i.e., there is a constant $\nu > 0$ so that

$$\nu \operatorname{vol} X_t \geq \sum_{i \in t} \operatorname{vol} X_i; \tag{4.15}$$

see also (1.20). It is known (cf. [126, Thm. 8.8.1]) that under the above assumptions there are constants $0 < c_{\mathscr{J},1} \leq c_{\mathscr{J},2}$ (independent of h and n) such that

$$c_{\mathscr{J},1}\|x\|_h \leq \|\mathscr{J} x\|_{L^2(\Omega)} \leq c_{\mathscr{J},2}\|x\|_h \quad \text{for all } x \in \mathbb{R}^I, \tag{4.16}$$

where $\mathscr{J}$ is defined in (4.9) and $\|x\|_h := \sqrt{h^d \sum_{i \in I} x_i^2}$ is induced by $(\cdot,\cdot)_h$ from (4.10).

From Theorem 4.10 it can be seen that the entries of inverses of symmetric positive definite sparse matrices decay exponentially. The mass matrix M is by definition symmetric positive definite and $(i,j) \in G(M)$, where $G(M)$ denotes the matrix graph of M defined in (1.11), implies that $X_i \cap X_j$ contains an interior point. Hence, if k is the smallest integer so that $\operatorname{dist}(X_i, X_j) \leq (k-1)h$, the length of a path in $G(M)$ from i to j must be at least k; i.e., $d_{ij} \geq 1 + e_{ij}/h$, where $e_{ij} := \operatorname{dist}(X_i, X_j)$.

Lemma 4.11. *Let $c_{\mathscr{J},1}$, $c_{\mathscr{J},2}$ be the constants from* (4.16). *Then*

$$|(M^{-1})_{ij}| \leq 2\|M^{-1}\|_2 \, \lambda^{e_{ij}/h+1} \quad \textit{for all } i,j \in I,$$

where $\lambda = \frac{\sqrt{r}-1}{\sqrt{r}+1} \in (0,1)$ with $r = (c_{\mathscr{J},2}/c_{\mathscr{J},1})^2$ do depend neither on h nor on the matrix size n.

Proof. Since $M = \mathscr{J}^* \mathscr{J}$, the condition number of M is bounded independently of the matrix size n by $(c_{\mathscr{J},2}/c_{\mathscr{J},1})^2$. Applying Theorem 4.10 and using $d_{ij} \geq 1 + e_{ij}/h$, we end up with the assertion. □

For the proof of the following theorem we will require that (4.11) holds for the maximum of the diameters of X_t and X_s instead of the minimum; i.e.,

$$\max\{\operatorname{diam} X_t, \operatorname{diam} X_s\} \leq \eta \operatorname{dist}(X_t, X_s). \tag{4.17}$$

Since the clusters t and s stem from the same level of a cluster tree, the minimum can be replaced by the maximum due to (1.22) if η is changed by a multiplicative constant $c > 1$ which will be omitted in the sequel.

Theorem 4.12. *For any $\varepsilon > 0$ there is $N_{\mathscr{H}} \in \mathscr{H}(T_{I\times I}, k)$ satisfying*

$$\|M^{-1} - N_{\mathscr{H}}\|_2 \le \varepsilon \|M^{-1}\|_2,$$

where $k \sim \max\{L^d, |\log \varepsilon|^d\}$ and $L = L(T_{I\times I})$ denotes the depth of the tree $T_{I\times I}$.

Proof. Let $k \in \mathbb{N}$, $k \ge n_{\min}$, and $b = t \times s \in P$. We set

$$(N_{\mathscr{H}})_b := \begin{cases} (M^{-1})_b, & \text{if } \min\{|t|,|s|\} \le k, \\ 0, & \text{else.} \end{cases}$$

Then $N_{\mathscr{H}}$ belongs to $\mathscr{H}(T_{I\times I}, k)$, since either $\operatorname{rank}(N_{\mathscr{H}})_b \le \min\{|t|,|s|\} \le k$ or $\operatorname{rank}(N_{\mathscr{H}})_b = 0$ for all blocks in P satisfying $\min\{|t|,|s|\} > k$.

Let $E = M^{-1} - N_{\mathscr{H}}$ be the error matrix. Due to Theorem 2.16 it remains to determine the spectral norm of $E_b = (M^{-1})_b$ for blocks $b = t \times s$ satisfying $\min\{|t|,|s|\} > k$. Then b is admissible because $\min\{|t|,|s|\} > k \ge n_{\min}$. Condition (4.17) implies for $(i,j) \in b$ that

$$e_{ij} = \operatorname{dist}(X_i, X_j) \ge \operatorname{dist}(X_t, X_s) \ge \eta^{-1} \max\{\operatorname{diam} X_t, \operatorname{diam} X_s\}.$$

From (4.14) and (4.15) we obtain

$$\operatorname{vol} X_t \ge \frac{1}{\nu} \sum_{i\in t} \operatorname{vol} X_i \ge \frac{c_v}{\nu} h^d |t|.$$

Together with $(\operatorname{diam} X_t)^d \ge \operatorname{vol} X_t / \omega_d$ it follows that $e_{ij} \ge c' h \sqrt[d]{|t|}$ with $c' > 0$ expressed by ω_d, η, ν, and c_v. Similarly, $e_{ij} \ge c' h \sqrt[d]{|s|}$ holds. The combination of the last two estimates yields $e_{ij}/h \ge c' \sqrt[2d]{|t||s|}$, which according to Lemma 4.11 proves

$$|E_{ij}| \le c \|M^{-1}\|_2 \lambda^{c' \sqrt[2d]{|t||s|}}.$$

A trivial estimate of the spectral norm yields

$$\|E_b\|_2 \le \sqrt{|t||s|} \max_{i\in t, j\in s} |E_{ij}| \le c\sqrt{|t||s|}\, \|M^{-1}\|_2 \lambda^{c' \sqrt[2d]{|t||s|}}.$$

We simplify the right-hand side: For a suitable $0 < c'' < c'$ the estimate $cr\lambda^{c' \sqrt[d]{r}} \le \lambda^{2c'' \sqrt[d]{r}}$ holds for all r greater than some $k_{\min} \in \mathbb{N}$. Hence, for $\ell \le L$

$$\|E_b\|_2 \le \|M^{-1}\|_2 \lambda^{2c'' \sqrt[2d]{|t||s|}} \le \|M^{-1}\|_2 \lambda^{2c'' \sqrt[d]{k}} \le 2^{-\ell} \|M^{-1}\|_2 \lambda^{c'' \sqrt[d]{k}}$$

if we assume that $k \ge k'_{\min} := [1/c'' \log_{1/\lambda} 2]^d L^d$, which gives $\lambda^{c'' \sqrt[d]{k}} \le 2^{-\ell}$. From the proof of Theorem 2.16 one has that

$$\|E\|_2 \le c_{\mathrm{sp}} \sum_{\ell=0}^{L} 2^{-\ell} \|M^{-1}\|_2 \lambda^{c'' \sqrt[d]{k}} \le 2c_{\mathrm{sp}} \|M^{-1}\|_2 \lambda^{c'' \sqrt[d]{k}}.$$

Choose $k \ge \max\{k_{\min}, k'_{\min}\}$ such that $2c_{\mathrm{sp}}\lambda^{c'' \sqrt[d]{k}} \le \varepsilon$. The last conditions are satisfied for $k \sim \max\{L^d, |\log \varepsilon|^d\}$. □

The proof of the last theorem cannot be applied to FE stiffness matrices since the decay rate of the inverse's entries depends on n. In the next section we will generalize the previous result to partitions constructed using an algebraic admissibility condition. We remark that in this situation it is not required that the discretization is quasi-uniform or shape regular.

4.1.3 An Algebraic Approach to the Approximation of the Inverse

In this section we make a first algebraic approach to the approximation of sparse matrices by $\mathscr{H}$-matrices; cf. [25]. We assume that A is sparse in the sense that (1.33) is valid.

We define the **essential boundary** of s with respect to t as

$$\partial_t s := \{\nu \in s : \exists i \in t \text{ such that } d_{i\nu} = \min_{j \in s} d_{ij}\}.$$

Analogously,

$$\partial_s t := \{\nu \in t : \exists j \in s \text{ such that } d_{\nu j} = \min_{i \in t} d_{ij}\}$$

is the essential boundary of t with respect to s; see Fig. 4.1.

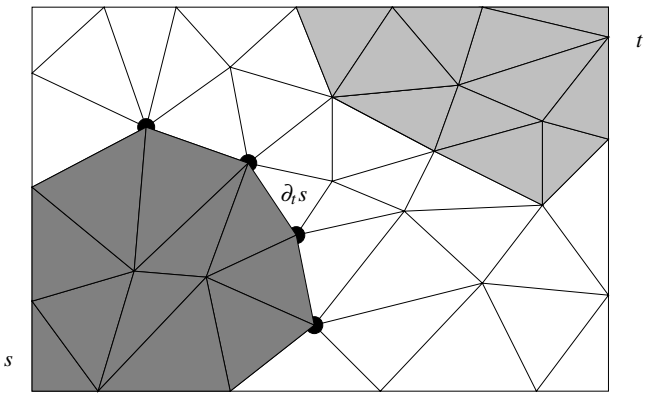


Fig. 4.1 Two clusters t and s with $\partial_t s$.

In the following lemma it will be seen that the rank of restrictions $[p(A)]_{ts}$ of matrix polynomials to blocks $t \times s$ with bounded ratio of $\operatorname{diam} \partial_t s$ and $\operatorname{dist}(t,s)$ can be estimated by the polynomial degree.

Lemma 4.13. *Let $t, s \subset I$ satisfy $\operatorname{diam} \partial_t s \le \eta \operatorname{dist}(t,s)$ for some $\eta > 0$. Then for $p \in \Pi_k$ it holds that $\operatorname{rank}[p(A)]_{ts} \le c_V (1+\eta)^m k^m$. Furthermore, $\operatorname{rank}[p(A^H A) A^H]_{ts} \le c_V (1+\eta)^m (2k+1)^m$.*

Proof. Let $\ell \le k$ and consider $i \in L_\mu := \{i \in t : \operatorname{dist}(i, \partial_t s) = \mu\}$ for $\mu > \ell$. Each minimal path from i to an arbitrary $j \in s$ includes an index $\nu \in \partial_t s$. Hence,

$$d_{ij} = d_{i\nu} + d_{\nu j} \ge \mu > \ell.$$

From Lemma 4.9 it follows that $(A^\ell)_{ij} = 0$. Since $t = \bigcup_{\mu=\operatorname{dist}(t,s)}^{\infty} L_\mu$, only the rows

$$N := \bigcup_{\mu=\operatorname{dist}(t,s)}^{\ell} L_\mu = \{i \in t : \operatorname{dist}(t,s) \le \operatorname{dist}(i, \partial_t s) \le \ell\}$$

of $(A^\ell)_{ts}$ do not vanish.

We may assume that $k \ge \operatorname{dist}(t,s)$ because the last set is empty if $k < \operatorname{dist}(t,s)$. According to assumption (1.33), we have that

$$|N| \le c_V (k + \operatorname{diam} \partial_t s)^m \le c_V (k + \eta \operatorname{dist}(t,s))^m \le c_V (1+\eta)^m k^m,$$

which proves the assertion. □

Remark 4.14. Assume for a moment that $\operatorname{diam} \partial_t s$ is bounded. From the last proof it can be seen that in this case the condition $\operatorname{diam} \partial_t s \le \eta \operatorname{dist}(t,s)$ can be omitted. Since we may interchange the roles of t and s, it is also sufficient that $\operatorname{diam} \partial_s t$ is bounded. In either case, we still obtain the result $\operatorname{rank}[p(A)]_{ts} \sim k^m$.

Let $T^*_{I\times I}$ be a block cluster tree with leaves $b = t \times s$ such that one of the following four conditions is satisfied:

(i) $\operatorname{diam} \partial_s t$ is bounded;
(ii) $\operatorname{diam} \partial_t s$ is bounded;
(iii) $\min\{\operatorname{diam} \partial_s t, \operatorname{diam} \partial_t s\} \le \eta \operatorname{dist}(t,s)$ for some parameter $\eta > 0$;
(iv) $\min\{|t|, |s|\} \le n_{\min}$ for some given number $n_{\min} \in \mathbb{N}$.

In Example 1.38 we have proved that the sparsity constant of partitions satisfying the previous conditions is bounded by a constant.

In [134] it was observed that the usual admissibility condition (4.11) can be relaxed by a so-called **weak admissibility**. Although progress has been made, this phenomenon is not yet fully understood. Note that for clusters t and s sharing a single corner point we have that $\operatorname{diam} \partial_t s = 1$. Weakly admissible partitions are hence covered by the above admissibility condition (i)–(iv).

The following theorem lays ground to the approximation of the inverse of a sparse matrix A by a hierarchical matrix on the partition $\mathscr{L}(T^*_{I\times I})$. In Theorem 4.15

we consider general non-singular matrices A, and in Theorem 4.16 positive definite matrices will be treated.

Theorem 4.15. *Let A be a non-singular matrix satisfying* (1.33). *Then there is $C_{\mathscr{H}} \in \mathscr{H}(T^*_{I\times I}, k)$ with $k \sim (|\log\varepsilon|/|\log\lambda|)^m$ such that*

$$\|A^{-1} - C_{\mathscr{H}}\|_2 \le \mathrm{cond}_2(A)\,\varepsilon\|A^{-1}\|_2,$$

where

$$\lambda = 1 - \frac{1}{(\mathrm{cond}_2(A))^2}.$$

Proof. Since A is non-singular, the matrix $B := A^H A$ has positive eigenvalues. Setting $\beta := 1/\lambda_{\max}(B)$, it follows that $\rho(I-\beta B) = 1 - \lambda_{\min}(B)/\lambda_{\max}(B) < 1$. The Neumann series applied to $I-\beta B$ gives the following representation of the inverse of B

$$B^{-1} = \beta \sum_{j=0}^{\infty} (I-\beta B)^j.$$

Setting

$$p_k(B) := \beta \sum_{j=0}^{k-1} (I-\beta B)^j,$$

we obtain that

$$\|B^{-1} - p_k(B)\|_2 \le \beta \sum_{j=k}^{\infty} \rho^j(I-\beta B) \le \frac{\beta\rho^k(I-\beta B)}{1-\rho(I-\beta B)} = \left(1 - \frac{\lambda_{\min}(B)}{\lambda_{\max}(B)}\right)^k \|B^{-1}\|_2$$

and hence

$$\begin{aligned}
\|A^{-1} - p_k(B)A^H\|_2 = \|(B^{-1} - p_k(B))A^H\|_2 &\le \|B^{-1} - p_k(B)\|_2\|A\|_2 \\
&\le \left(1 - \frac{\lambda_{\min}(B)}{\lambda_{\max}(B)}\right)^k \|B^{-1}\|_2\|A\|_2 \\
&= \mathrm{cond}_2(A)\left(1 - \frac{1}{(\mathrm{cond}_2(A))^2}\right)^k \|A^{-1}\|_2.
\end{aligned}$$

The last lemma proves that each block of $C_{\mathscr{H}} := p_k(B)A^H$ satisfying one of the conditions (i)–(iv) has rank at most $c_V(1+\eta)^m(2k+1)^m$. □

Theorem 4.16. *Let A be a symmetric positive definite matrix satisfying* (1.33). *Then there is $C_{\mathscr{H}} \in \mathscr{H}(T^*_{I\times I}, k)$ with $k \sim (|\log\varepsilon|/|\log\lambda|)^m$ such that*

$$\|A^{-1} - C_{\mathscr{H}}\|_2 \le \varepsilon\|A^{-1}\|_2,$$

where

$$\lambda = \frac{\sqrt{\mathrm{cond}_2(A)} - 1}{\sqrt{\mathrm{cond}_2(A)} + 1}.$$

Proof. We use similar arguments as in the proof of Theorem 4.10. For all algebraic polynomials p it holds that

$$\|A^{-1} - p(A)\|_2 = \rho(A^{-1} - p(A)) = \max_{x \in \sigma(A)} |x^{-1} - p(x)|.$$

Furthermore, Π_k contains p_k so that

$$\|x^{-1} - p_k(x)\|_{\infty,\sigma(A)} \leq c\,\lambda^{k+1},$$

where

$$c = \frac{(1+\sqrt{r})^2}{2r}\|A^{-1}\|_2 \leq 2\|A^{-1}\|_2, \quad \lambda = \frac{\sqrt{r}-1}{\sqrt{r}+1}, \quad r = \mathrm{cond}_2(A).$$

Let $C_{\mathscr{H}} := p_k(A)$. Lemma 4.13 proves that each block of $C_{\mathscr{H}} := p_k(A)$ satisfying one of the conditions (i)–(iv) has rank at most $c_V(1+\eta)^m k^m$. □

The previous theorems show that the geometric condition (4.11) is not necessary. Hence, cardinality balanced clustering is enough, which implies that $\mathscr{H}$-matrices can be applied to arbitrary grids.

Although we were able to generalize the partition on which approximation is possible, the convergence rate λ goes to 1 if $\mathrm{cond}_2(A)$ goes to infinity. The last theorems are therefore meaningful only for well-conditioned matrices. The numerical experiments from Sect. 4.5, however, show that the actual rank k is bounded even if $\mathrm{cond}_2(A) \to \infty$ for $n \to \infty$.

In order to prove rigorously that the inverse of finite element stiffness matrices can be approximated by $\mathscr{H}$-matrices, we have to use a different technique [27, 20]. For this purpose, we leave our algebraic point of view and exploit analytic properties of elliptic operators. The following proof is based on the approximation of the Green's function.

4.1.4 Degenerate Approximation of the Green's Function

In the last decade considerable attention has been paid to fast algorithms for the direct solution of boundary value problems. If Ω is an interval, then the Green function G of ordinary differential operators has the property that

$$G(x,y) = \begin{cases} u_1(x)v_1(y), & x \geq y, \\ u_2(x)v_2(y), & x \leq y \end{cases}$$

with appropriately chosen functions u_1, v_1, u_2, and v_2; see the following Example 4.17. This fact is, for instance, exploited by the algorithm in [239].

Example 4.17. The solution of the two-point boundary value problem

$$-u''(x) = f(x), \quad x \in \Omega := (0,1),$$
$$u(0) = u(1) = 1$$

can be represented by

$$u(x) = \int_\Omega G(x,y) f(y)\,\mathrm{d}y,$$

where

$$G(x,y) = \begin{cases} (1-x)y, & 0 \le y \le x, \\ x(1-y), & x \le y \le 1 \end{cases}$$

denotes the Green's function. The restriction of G to each pair of domains $D_1 \times D_2$ satisfying $\operatorname{dist}(D_1, D_2) > 0$ is the product of two functions v and w

$$G(x,y) = v(x)w(y), \quad (x,y) \in D_1 \times D_2.$$

The algebraic analogue of the above property are semi-separable matrices, which have been introduced in Sect. 1.2. The exact representation of G as a degenerate function cannot be carried over to problems of spatial dimension $d \ge 2$.

The aim of this section is to show that the Green's function corresponding to the operator (4.2) and the domain Ω can be approximated by a degenerate kernel

$$G_k(x,y) = \sum_{i=1}^{k} u_i(x) v_i(y) \tag{4.18}$$

with appropriate functions u_i, v_i, $i = 1,\ldots,k$, on admissible pairs (D_1, D_2) of subdomains of Ω, i.e., on domains $D_1, D_2 \subset \Omega$ satisfying

$$\min\{\operatorname{diam} D_1, \operatorname{diam} D_2\} \le \eta \operatorname{dist}(D_1, D_2).$$

According to the theorem of De Giorgi (see [104, p. 200]), G is locally only Hölder continuous if the coefficients of the operator are in L^∞. Hence, G cannot be approximated by polynomials. We have to find a different system of approximants. For this purpose a finite-dimensional space will be constructed which G is approximated from with respect to one variable. Before we turn to the construction of the approximation space, let us first state some properties of G.

Denote by $\mathscr{L}_0$ the principal part of $\mathscr{L}$ from (4.2), i.e., the special case $\beta_i = \gamma_i = \delta = 0$, $i = 1,\ldots,d$. For operators of type $\mathscr{L}_0$ it is shown in [124] that in the case $d \ge 3$ a Green's function $G_0 : \Omega \times \Omega \to \mathbb{R} \cup \{\infty\}$ exists with the properties

$$G_0(x,\cdot) \in H^1(\Omega \setminus B_r(x)) \cap W_0^{1,1}(\Omega) \quad \text{for all } x \in \Omega \text{ and all } r > 0,$$
$$a_0(G_0(x,\cdot), \varphi) = \varphi(x) \quad \text{for all } \varphi \in C_0^\infty(\Omega) \text{ and } x \in \Omega,$$

where $B_r(x)$ is the open ball centered at x with radius r and

$$a_0(u,v) = \int_\Omega \sum_{i,j=1}^{d} c_{ij}\partial_j u\, \partial_i v \,\mathrm{d}x. \tag{4.19}$$

Furthermore, for $x, y \in \Omega$ it holds that

$$|G_0(x,y)| \leq \frac{c_d(\Lambda_{\mathscr{L}}/\lambda_{\mathscr{L}})}{\lambda_{\mathscr{L}}} \|x-y\|^{2-d} \tag{4.20}$$

with a constant c depending on the diameter of Ω, d, and on the ratio of $\Lambda_{\mathscr{L}}$ and $\lambda_{\mathscr{L}}$. Note that a Green's function contains information about the domain Ω, while singularity functions (as defined in Chap. 3) depend on the operator $\mathscr{L}$ only. Systems of elliptic partial differential operators were investigated in [84]. Up to our knowledge, estimates of type (4.20) have not been derived for general operators $\mathscr{L}$ of type (4.2) in the presence of lower-order terms. We can however define a Green's function $G := (\mathscr{L}_0^{-1}\mathscr{L})^{-1}G_0$ satisfying

$$\begin{aligned} &G(x,\cdot) \in H^1(\Omega \setminus B_r(x)) \cap W_0^{1,1}(\Omega) && \text{for all } x \in \Omega \text{ and all } r > 0, \\ &a(G(x,\cdot),\varphi) = \varphi(x) && \text{for all } \varphi \in C_0^\infty(\Omega) \text{ and } x \in \Omega, \end{aligned}$$

where a is the bilinear form defined in (4.6). Notice that

$$G - G_0 = [(\mathscr{L}_0^{-1}\mathscr{L})^{-1} - \mathscr{I}]G_0 = -\mathscr{L}^{-1}\mathscr{L}_1 G_0,$$

where $\mathscr{L}_1 := \mathscr{L} - \mathscr{L}_0$ denotes the lower-order part of $\mathscr{L}$. Since $\mathscr{L}^{-1}\mathscr{L}_1$ is an operator of order -1, $\mathscr{L}^{-1}\mathscr{L}_1 G_0$ is smoother than G_0. Hence, the singularity of G_0 at $x = y$ is carried over to G, and we may assume that there is a constant c_d such that for $x, y \in \Omega$ it holds that

$$|G(x,y)| \leq \frac{c_d(\Lambda_{\mathscr{L}}/\lambda_{\mathscr{L}}, \beta, \gamma, \delta)}{\lambda_{\mathscr{L}}} \|x-y\|^{2-d}. \tag{4.21}$$

For $d = 2$, the existence of a Green's function for symmetric operators of type (4.2) has been proved in [84]. Instead of (4.21) one has the following bound on the Green's function

$$|G(x,y)| \leq \frac{c(\Lambda_{\mathscr{L}}/\lambda_{\mathscr{L}}, \beta, \gamma, \delta)}{\lambda_{\mathscr{L}}} \log \|x-y\| \tag{4.22}$$

for $x, y \in \Omega$.

4.1.4.1 A Caccioppoli Inequality

The whole theory of the approximation of discrete elliptic operators by hierarchical matrices can be founded on one single principle, the **interior regularity**, which is characteristic for elliptic operators. In Chap. 3 we have used the interior regularity expressed by a Caccioppoli-type inequality (see Lemma 3.3) for the proof of

the asymptotic smoothness of the singularity function. In this section we derive a Caccioppoli inequality for nonsmooth coefficients and apply it to the Green function of $\mathscr{L}$.

Let $D \subset \mathbb{R}^d$ be a domain satisfying $D \cap \Omega \neq 0$. The set

$$X(D) := \{u \in H^1_{\mathrm{loc}}(D) : a(u,\varphi) = 0 \text{ for all } \varphi \in C_0^\infty(D \cap \Omega) \text{ and } u|_{D\setminus\Omega} = 0\}, \quad (4.23)$$

consists of $\mathscr{L}$-harmonic H^1_{loc}-functions vanishing outside of Ω. Here and in the following we use the notation

$$H^1_{\mathrm{loc}}(D) := \{u \in L^2(D) : u \in H^1(K) \text{ for all compact } K \subset D\}.$$

The following Caccioppoli inequality (4.24) estimates the derivatives of harmonic solutions on a subdomain by its values on an enclosing domain. This provides a means to overcome the lack of regularity of G. Note that the coefficient function β, γ, and δ enter the estimate through the norm

$$\|\cdot\| := \|\,|\cdot|_2\|_{L^\infty}.$$

Hence, the quality of the following estimate depends on the size of the coefficients but not on their variation.

Lemma 4.18. *Let $K \subset D$ be compact. There is $c_{\mathscr{L}} = c_{\mathscr{L}}(\Lambda_{\mathscr{L}}/\lambda_{\mathscr{L}}, \lambda_{\mathscr{L}}, \beta, \gamma, \delta, \operatorname{diam} D)$ such that*

$$\|\nabla u\|_{L^2(K)} \le \frac{c_{\mathscr{L}}}{\operatorname{dist}(K, \partial D)} \|u\|_{L^2(D)} \quad (4.24)$$

for all $u \in X(D)$.

Proof. Let $\eta \in C^1(D)$ satisfy $0 \le \eta \le 1$, $\eta = 1$ in K, $\eta = 0$ in a neighborhood of ∂D, and $\|\nabla \eta\| \le 2/\sigma$ in D, where we set $\sigma = \operatorname{dist}(K, \partial D)$. Since $K' := \operatorname{supp} \eta$ is a compact subset of D, the definition (4.23) of $X(D)$ implies $u \in H^1(K')$. Hence, $\varphi := \eta^2 u \in H^1_0(D \cap \Omega)$ may be used as a test function in $a(u,\varphi) = 0$ due to the dense embedding of $C_0^\infty(D \cap \Omega)$ in $H^1_0(D \cap \Omega)$. Since $\varphi = 0$ in $\Omega \setminus D$, we have

$$\begin{aligned} &2\int_D \eta u \nabla \eta \cdot C \nabla u \,\mathrm{d}x + \int_D \eta^2 \nabla u \cdot C \nabla u \,\mathrm{d}x = \int_D \nabla(\eta^2 u) \cdot C \nabla u \,\mathrm{d}x \\ &= -\int_D \eta^2 u \beta \cdot \nabla u \,\mathrm{d}x - \int_D u \gamma \cdot \nabla(\eta^2 u) \,\mathrm{d}x - \int_D \delta \eta^2 |u|^2 \,\mathrm{d}x. \end{aligned}$$

Since C is positive definite, its square root $C^{1/2}$ is defined. Hence,

$$\begin{aligned} \int_D \eta^2 |C^{1/2} \nabla u|^2 \,\mathrm{d}x = &-2\int_D \eta u \nabla \eta \cdot C \nabla u \,\mathrm{d}x - \int_D \eta^2 u \beta \cdot \nabla u \,\mathrm{d}x \\ &-2\int_D \eta |u|^2 \gamma \cdot \nabla \eta \,\mathrm{d}x - \int_D \eta^2 u \gamma \cdot \nabla u \,\mathrm{d}x - \int_D \delta \eta^2 |u|^2 \,\mathrm{d}x. \end{aligned}$$

For the first integral on the right-hand side of the last equation we obtain

$$\left|\int_D \eta u \nabla\eta \cdot C\nabla u \,\mathrm{d}x\right| \le \int_D |C^{1/2}\nabla\eta|\,|\eta C^{1/2}\nabla u|\,|u|\,\mathrm{d}x$$
$$\le 2\frac{\sqrt{\Lambda_{\mathscr{L}}}}{\sigma}\|\eta C^{1/2}\nabla u\|_{L^2(D)}\|u\|_{L^2(D)}.$$

The second integral can be estimated as

$$\left|\int_D \eta^2 u\beta\cdot\nabla u\,\mathrm{d}x\right| \le \int_D \eta|\beta|\,\eta\|\nabla u\|\,|u|\,\mathrm{d}x \le \|\beta\|\int_D \eta\|\nabla u\|\,|u|\,\mathrm{d}x$$
$$\le \|\beta\|\left(\int_D \eta^2\|\nabla u\|^2\,\mathrm{d}x\right)^{1/2}\|u\|_{L^2(D)}$$
$$\le \frac{\|\beta\|}{\sqrt{\lambda_{\mathscr{L}}}}\|\eta C^{1/2}\nabla u\|_{L^2(D)}\|u\|_{L^2(D)}$$

and, similarly, one has for the forth integral

$$\left|\int_D \eta^2 u\gamma\cdot\nabla u\,\mathrm{d}x\right| \le \frac{\|\gamma\|}{\sqrt{\lambda_{\mathscr{L}}}}\|\eta C^{1/2}\nabla u\|_{L^2(D)}\|u\|_{L^2(D)}.$$

Since

$$2\|\eta C^{1/2}\nabla u\|_{L^2(D)}\|u\|_{L^2(D)} \le \frac{1}{\varepsilon}\|\eta C^{1/2}\nabla u\|^2_{L^2(D)} + \varepsilon\|u\|^2_{L^2(D)}$$

with $\varepsilon := 4\sqrt{\Lambda_{\mathscr{L}}}/\sigma + \lambda_{\mathscr{L}}^{-1/2}(\|\beta\|+\|\gamma\|)$, one readily checks that

$$\|\eta C^{1/2}\nabla u\|^2_{L^2(D)} \le 2\left(\frac{\varepsilon^2}{2} + \frac{4}{\sigma}\|\beta\| + \|\delta\|\right)\|u\|^2_{L^2(D)}.$$

This leads to

$$\|\eta C^{1/2}\nabla u\|_{L^2(D)} \le \frac{c}{\sigma}\|u\|_{L^2(D)},$$

where

$$c^2 = \left(4\sqrt{\Lambda_{\mathscr{L}}} + \frac{\sigma}{\sqrt{\lambda_{\mathscr{L}}}}(\|\beta\|+\|\gamma\|)\right)^2 + 8\sigma\|\beta\| + 2\sigma^2\|\delta\|,$$

and hence

$$\|\nabla u\|_{L^2(K)} \le \|\eta\nabla u\|_{L^2(D)} \le \lambda_{\mathscr{L}}^{-1/2}\|\eta C^{1/2}\nabla u\|_{L^2(D)} \le \frac{c}{\sigma\sqrt{\lambda_{\mathscr{L}}}}\|u\|_{L^2(D)}.$$

The rough estimate $\sigma \le \operatorname{diam} D$ proves the assertion. □

Remark 4.19. From the preceding proof it can be seen that the coefficient δ does not enter estimate (4.24) if $\delta \ge 0$.

When we try to generalize the above properties to systems of partial differential equations, we find the principle difficulty that a Caccioppoli inequality does not hold for general nonsmooth coefficients; see the counter example in [105]. Note that in

Lemma 3.3 we have proved a Caccioppoli inequality for systems with smooth coefficients, which could be used to prove the existence of $\mathscr{H}$-matrix approximants to the inverse FE stiffness matrix in the same way as we proceed now for scalar problems. Caccioppoli inequalities for the operators $\operatorname{curl}\alpha\operatorname{curl}+\beta\mathscr{I}$ and $\operatorname{curl}\alpha\operatorname{curl}-\beta\nabla\operatorname{div}$ with nonsmooth coefficients α,β are derived in [31].

4.1.4.2 Construction of the Approximation Space

In the rest of this section $D\subset\mathbb{R}^d$ is an arbitrary domain. The proof of the following basic lemma is mainly based on the Poincaré inequality; cf. [104].

Lemma 4.20. *Let D be convex and X a closed subspace of $L^2(D)$. Then for any $k\in\mathbb{N}$ there is a subspace $V_k\subset X$ satisfying $\dim V_k\le k$ so that*

$$\operatorname{dist}_{L^2(D)}(u,V_k)\le c_A\frac{\operatorname{diam}D}{\sqrt[d]{k}}\|\nabla u\|_{L^2(D)} \tag{4.25}$$

for each $u\in X\cap H^1(D)$, where c_A depends only on the spatial dimension d.

Proof. First we assume $k=\ell^d$ and $D\subset Q:=\{x\in\mathbb{R}^d:\|x-z\|_\infty<\frac{1}{2}\operatorname{diam}D\}$ for some $z\in\mathbb{R}^d$. We subdivide the cube Q uniformly into k sub-cubes Q_i, $i=1,\dots,k$, and set $D_i=D\cap Q_i$, $i=1,\dots,k$. Each of the sets D_i is convex with $\operatorname{diam}D_i\le\frac{\sqrt{d}}{\ell}\operatorname{diam}D$. Let

$$W_k=\{v\in L^2(D):v\text{ is constant on each }D_i,\,i=1,\dots,k\}.$$

Then $\dim W_k\le k$ and according to Poincaré's inequality for $u\in H^1(D)$ (in particular, we use the convex version in [200, 19] with explicitly given constant) it holds that

$$\int_{D_i}|u-\bar u_i|^2\,\mathrm{d}x\le\pi^{-2}(\operatorname{diam}D_i)^2\int_{D_i}\|\nabla u\|^2\,\mathrm{d}x,$$

where $\bar u_i=(\operatorname{vol}D_i)^{-1}\int_{D_i}u\,\mathrm{d}x$ is the mean value of u in D_i. Summation over all i yields

$$\|u-\bar u\|_{L^2(D)}\le\frac{\sqrt{d}}{\pi\ell}\operatorname{diam}D\,\|\nabla u\|_{L^2(D)}$$

for $\bar u\in W_k$ defined by $\bar u|_{D_i}=\bar u_i$.

For general $k\in\mathbb{N}$ choose $\ell:=\lfloor\sqrt[d]{k}\rfloor\in\mathbb{N}$, i.e., $\ell^d\le k<(\ell+1)^d$. Applying the previous arguments to $k':=\ell^d$, we obtain the space $W_k:=W_{k'}$ satisfying $\dim W_k=\dim W_{k'}\le k'\le k$. Using $\ell\ge(\ell+1)/2>\sqrt[d]{k}/2$, we arrive at

$$\|u-\bar u\|_{L^2(D)}\le c_A\frac{\operatorname{diam}D}{\sqrt[d]{k}}\,\|\nabla u\|_{L^2(D)}$$

with the constant $c_A:=2\sqrt{d}/\pi$.

In order to guarantee that the approximation is done from a subset of X, we project W_k onto X. Let $P : L^2(D) \to X$ be the $L^2(D)$-orthogonal projection onto X and let $V_k = P(W_k)$. Keeping in mind that P has norm one and $u \in X$, we obtain

$$\operatorname{dist}_{L^2(D)}(u, V_k) \le \|u - P\bar{u}\|_{L^2(D)} = \|P(u - \bar{u})\|_{L^2(D)} \le \|u - \bar{u}\|_{L^2(D)},$$

which proves the assertion. □

In order to be able to use $X(D)$ from (4.23) as X in Lemma 4.20, in the following lemma it is proved that $X(D)$ is a closed subspace of $L^2(D)$. We remark that the extension of $G(x,\cdot)$, $x \in \Omega$, to $\mathbb{R}^d$ by zero is in $X(D)$ for all $D \subset \mathbb{R}^d$ satisfying $\operatorname{dist}(x, D) > 0$.

Lemma 4.21. *The space $X(D)$ is closed in $L^2(D)$.*

Proof. Let $\{u_k\}_{k\in\mathbb{N}} \subset X(D)$ converge to u in $L^2(D)$ and let $K \subset D$ be a compact subset such that $\operatorname{dist}(K, \partial D) > 0$. According to Lemma 4.18, the sequence $\{\nabla u_k\}_{k\in\mathbb{N}}$ is bounded on K,

$$\|\nabla u_k\|_{L^2(K)} \le c\,\|u_k\|_{L^2(D)} \le C.$$

Due to the Banach-Alaoglu Theorem, a subsequence $\{u_{i_k}\}_{k\in\mathbb{N}}$ converges weakly in $H^1(K)$ to $\hat{u} \in H^1(K)$. Hence, for any $v \in L^2(K)$ we have

$$(u, v)_{L^2(K)} = \lim_{k\to\infty} (u_{i_k}, v)_{L^2(K)} = (\hat{u}, v)_{L^2(K)},$$

which proves that $u = \hat{u} \in H^1(K)$. Since the functional $a(\cdot, \varphi)$ for $\varphi \in C_0^\infty(D)$ is in $(H^1(K))'$, we see by the same argument that $a(u, \varphi) = 0$. Finally, $u_k|_{D\setminus\Omega} = 0$ leads to $u|_{D\setminus\Omega} = 0$. Hence, $u \in X(D)$ is proved. □

4.1.4.3 Exponentially-Accurate Space of $\mathscr{L}$-Harmonic Functions

Lemma 4.22. *Assume that $D_2 \subset D$ is a convex domain such that for some $\eta > 0$ it holds that*

$$0 < \operatorname{diam} D_2 \le \eta \operatorname{dist}(D_2, \partial D).$$

Then for any $\varepsilon > 0$ there is a subspace $W \subset X(D_2)$ so that

$$\operatorname{dist}_{L^2(D_2)}(u, W) \le \varepsilon \|u\|_{L^2(D)} \quad \text{for all } u \in X(D) \tag{4.26}$$

and $\dim W \le c_\eta^d \lceil |\log \varepsilon| \rceil^{d+1} + \lceil |\log \varepsilon| \rceil$, *where* $c_\eta := c_A c_{\mathscr{L}} e(2+\eta)$.

Proof. Let $\ell := \lceil |\log \varepsilon| \rceil$. We consider a nested sequence of convex domains

$$K_j = \{x \in \mathbb{R}^d : \operatorname{dist}(x, D_2) \le r_j\}$$

with real numbers $r_j := (1 - j/\ell)\operatorname{dist}(D_2, \partial D)$, $j = 0, \ldots, \ell$. Notice that

$$D_2 = K_\ell \subset K_{\ell-1} \subset \cdots \subset K_0 \subset D.$$

Using the definition (4.23) of the space X, we set $X_j := X(K_j)$.

Applying Lemma 4.20 to K_j with the choice $k := \lceil (c_A c_{\mathscr{L}}(2+\eta)\ell\varepsilon^{-1/\ell})^d \rceil$, we can find a subspace $V_j \subset X_j$ satisfying $\dim V_j \le k$ and

$$\operatorname{dist}_{L^2(K_j)}(v, V_j) \le c_A \frac{\operatorname{diam} K_j}{\sqrt[d]{k}} \|\nabla v\|_{L^2(K_j)} \tag{4.27}$$

for all $v \in X_j \cap H^1(K_j)$. From Lemma 4.18 applied to (K_j, K_{j-1}) instead of (K, D), we obtain

$$\|\nabla v\|_{L^2(K_j)} \le \frac{c_{\mathscr{L}}}{\operatorname{dist}(K_j, \partial K_{j-1})} \|v\|_{L^2(K_{j-1})} = c_{\mathscr{L}} \frac{\ell}{r_0} \|v\|_{L^2(K_{j-1})} \tag{4.28}$$

for all $v \in X_{j-1}$. Since any $v \in X_{j-1}$ also belongs to $X_j \cap H^1(K_j)$, the estimates (4.27) and (4.28) together with $\operatorname{diam} K_j \le (2+\eta) r_0$ may be combined to

$$\operatorname{dist}_{L^2(K_j)}(v, V_j) \le \varepsilon^{1/\ell} \|v\|_{L^2(K_{j-1})} \quad \text{for all } v \in X_{j-1}.$$

Let $u \in X(D)$ and $v_0 := u|_{K_0} \in X_0$. By the last estimate we have $v_0|_{K_1} = u_1 + v_1$ with $u_1 \in V_1$ and

$$\|v_1\|_{L^2(K_1)} \le \varepsilon^{1/\ell} \|v_0\|_{L^2(K_0)}.$$

Consequently, v_1 belongs to X_1. Similarly, for all $j = 1, \dots, \ell$, we are able to find an approximant $u_j \in V_j$ so that $v_{j-1}|_{K_j} = u_j + v_j$ and $\|v_j\|_{L^2(K_j)} \le \varepsilon^{1/\ell} \|v_{j-1}\|_{L^2(K_{j-1})}$. Using the restrictions of V_j to the smallest domain $D_2 = K_\ell$, let

$$W := \operatorname{span}\{V_j|_{D_2},\ j = 1, \dots, \ell\}.$$

Then W is a subspace of $X(D_2)$ and, since $v_0|_{D_2} = v_\ell + \sum_{j=1}^{\ell} u_j|_{D_2}$, we are led to

$$\operatorname{dist}_{L^2(D_2)}(v_0, W) \le \|v_\ell\|_{L^2(D_2)} \le \left(\varepsilon^{1/\ell}\right)^\ell \|v_0\|_{L^2(K_0)} \le \varepsilon \|u\|_{L^2(D)},$$

where the last inequality is due to $K_0 \subset D$.

The dimension of W is bounded by $\sum_{j=1}^{\ell} \dim V_j \le \ell k$. Since $\varepsilon^{-1/\ell} \le e$, we obtain $\dim W \le (c_A c_{\mathscr{L}} e(2+\eta))^d \ell^{d+1} + \ell$. □

Assume that $\operatorname{diam} D_2 \le \operatorname{diam} D_1$. The previous lemma will now be applied to the Green's functions $G(x,\cdot)$ on D_2 with $x \in D_1 \subset \Omega$. For this purpose let g_x be the extension of $G(x,\cdot)$ to $\mathbb{R}^d \setminus \overline{D}_1$; i.e.,

$$g_x(y) := \begin{cases} G(x,y), & y \in \Omega \setminus \overline{D}_1, \\ 0, & y \in \mathbb{R}^d \setminus \Omega. \end{cases}$$

Then g_x is in $X(\mathbb{R}^d \setminus \overline{D}_1)$. Note that its approximant $G_k(x,\cdot)$ from the following theorem is of the desired form (4.18).

Theorem 4.23. *Let $D_1 \subset \Omega$ and let $D_2 \subset \mathbb{R}^d$ be convex. Assume that there is $\eta > 0$ such that*

$$0 < \operatorname{diam} D_2 \le \eta \operatorname{dist}(D_1, D_2).$$

Then for any $\varepsilon > 0$ there is a separable approximation

$$G_k(x,y) = \sum_{i=1}^{k} u_i(x) v_i(y) \quad \text{with } k \le k_\varepsilon := c_\eta^d \lceil |\log \varepsilon| \rceil^{d+1} + \lceil |\log \varepsilon| \rceil,$$

so that for all $x \in D_1$

$$\|G(x,\cdot) - G_k(x,\cdot)\|_{L^2(D_2 \cap \Omega)} \le \varepsilon \|G(x,\cdot)\|_{L^2(\hat{D}_2)}, \tag{4.29}$$

where $\hat{D}_2 := \{y \in \Omega : 2\eta \operatorname{dist}(y, D_2) < \operatorname{diam} D_2\}$,

$$c_\eta = 2c_A e(1+\eta) \left(\left(4\sqrt{\Lambda_{\mathscr{L}}/\lambda_{\mathscr{L}}} + \frac{\sigma}{\lambda_{\mathscr{L}}} (\|\beta\| + \|\gamma\|) \right)^2 + 2\frac{\sigma}{\lambda_{\mathscr{L}}} (4\|\beta\| + \sigma\|\delta\|) \right)^{1/2},$$

and $\sigma := \operatorname{diam} D_2/(2\eta)$.

Proof. Let

$$D = \{y \in \mathbb{R}^d : 2\eta \operatorname{dist}(y, D_2) < \operatorname{diam} D_2\}.$$

Note that due to $\operatorname{dist}(D_1, D) > 0$, we have $g_x \in X(D)$ for all $x \in D_1$. Since in addition $\operatorname{diam} D_2 \le 2\eta \operatorname{dist}(D_2, \partial D)$, Lemma 4.22 can be applied with η replaced by 2η. Let $\{v_1, \ldots, v_k\}$ be a basis of the subspace $W \subset X(D_2)$ with

$$k = \dim W \le c_{2\eta}^d \lceil |\log \varepsilon| \rceil^{d+1} + \lceil |\log \varepsilon| \rceil.$$

According to (4.26), g_x can be decomposed into $g_x = \hat{g}_x + r_x$ with $\hat{g}_x \in W$ and $\|r_x\|_{L^2(D_2)} \le \varepsilon \|g_x\|_{L^2(D)}$. Since g_x and $\hat{g}_x$ vanish outside of Ω, we actually have $\|r_x\|_{L^2(D_2 \cap \Omega)} \le \varepsilon \|G(x,\cdot)\|_{L^2(\hat{D}_2)}$. Expressing $\hat{g}_x$ by means of the basis of W, we obtain

$$\hat{g}_x = \sum_{i=1}^{k} u_i(x) v_i$$

with coefficients $u_i(x)$ depending on the index $x \in D_1$. The function

$$G_k(x,y) := \sum_{i=1}^{k} u_i(x) v_i(y)$$

satisfies estimate (4.29). □

Remark 4.24. Without loss of generality, we may choose $\{v_1, \ldots, v_k\}$ as an orthogonal basis of W. Then the coefficients $u_i(x)$ in the latter expansion are the Fourier coefficients $(G(x,\cdot), v_i)_{L^2(D_2 \cap \Omega)}$ showing that u_i satisfies $\mathscr{L} u_i = v_i$ with homogeneous Dirichlet boundary conditions. In particular, u_i is $\mathscr{L}$-harmonic in $\Omega \setminus D_2$. Note that the u_i's do not depend on D_1.

Since the constant c_A does not depend on Ω, the geometry enters c_η only through the diameter. Hence, the shape of the domain does not influence the previous approximation result. Additionally, only the norm of the coefficients of the operator appear in the estimates.

4.1.5 Approximation of Discrete Operators

The existence of degenerate approximants to the Green's function from the last section will now be used to prove existence of $\mathscr{H}$-matrix approximants to the discrete inverse B of $\mathscr{L}$ from (4.8) and the inverse stiffness matrix A^{-1}.

Theorem 4.25. *Let X_t be convex for all $t \in T_I$. For any $\varepsilon > 0$ let $k_\varepsilon \in \mathbb{N}$ be chosen as in Theorem 4.23. Then for $k \geq \max\{k_\varepsilon, n_{\min}\}$ there is $B_{\mathscr{H}} \in \mathscr{H}(T_{I\times I}, k)$ such that*

$$\|B - B_{\mathscr{H}}\|_2 \leq c_d \frac{\varepsilon}{\lambda_{\mathscr{L}}}, \tag{4.30}$$

where $c_d = c_d(\Lambda_{\mathscr{L}}/\lambda_{\mathscr{L}}, \beta, \gamma, \delta, \eta, \operatorname{diam}\Omega)$.

Proof. Let $b = t \times s \in P$ with $\min\{|t|, |s|\} \leq n_{\min}$. In this case we simply set

$$(B_{\mathscr{H}})_b := B_b = (\mathscr{J}^* \mathscr{L}^{-1} \mathscr{J})_b.$$

Since the block $(B_{\mathscr{H}})_b$ has at most $n_{\min}$ columns or rows, $\operatorname{rank}(B_{\mathscr{H}})_b \leq k$ holds.

If $b = t \times s \in P$ with $\min\{|t|, |s|\} > n_{\min}$, then b satisfies (4.11). Applying Theorem 4.23 with $D_1 = X_t$, $D_2 = X_s$, there is $\tilde{G}_b(x,y) = \sum_{i=1}^{k_\varepsilon} u_i^b(x) v_i^b(y)$ such that

$$\|G - \tilde{G}_b\|_{L^2(X_t \times X_s)} \leq \varepsilon \|G\|_{L^2(X_t \times \hat{X}_s)},$$

where $\hat{X}_s := \{x \in \Omega : 2\eta \operatorname{dist}(x, X_s) \leq \operatorname{diam} X_s\}$. Let the functions u_i^b and v_i^b be extended to Ω by zero. We define the integral operator

$$\mathscr{K}_b \varphi = \int_\Omega \tilde{G}_b(x, \cdot) \varphi(x) \, \mathrm{d}x \quad \text{for supp } \varphi \subset \overline{\Omega}$$

and set $(B_{\mathscr{H}})_b = (\mathscr{J}^* \mathscr{K}_b \mathscr{J})_b$. The rank of $(B_{\mathscr{H}})_b$ is bounded by k_ε; see Sect. 3.3.

Let $x \in \mathbb{R}^s$ and $y \in \mathbb{R}^t$. To see that $(B_{\mathscr{H}})_b$ approximates the block B_b, remember the representation (4.12) of $\mathscr{L}^{-1}$ and use (4.16). The estimate

$$\begin{aligned}
((B - B_{\mathscr{H}})_b x, y)_h &= (\mathscr{J}^* (\mathscr{L}^{-1} - \mathscr{K}_b) \mathscr{J} x, y)_h = ((\mathscr{L}^{-1} - \mathscr{K}_b) \mathscr{J} x, \mathscr{J} y)_{L^2} \\
&\leq \|G - \tilde{G}_b\|_{L^2(X_t \times X_s)} \|\mathscr{J} x\|_{L^2(X_s)} \|\mathscr{J} y\|_{L^2(X_t)} \\
&\leq \varepsilon \|G\|_{L^2(X_t \times \hat{X}_s)} \|\mathscr{J} x\|_{L^2(\Omega)} \|\mathscr{J} y\|_{L^2(\Omega)} \\
&\leq \varepsilon c_{\mathscr{J},2}^2 \|G\|_{L^2(X_t \times \hat{X}_s)} \|x\|_h \|y\|_h
\end{aligned}$$

proves $\|(B - B_{\mathscr{H}})_b\|_2 \leq \varepsilon c_{\mathscr{J},2}^2 \|G\|_{L^2(X_t \times \hat{X}_s)}$.

Although $G(x,\cdot) \in W^{1,1}(\Omega)$ for all $x \in \Omega$, $G(\cdot,\cdot)$ does not belong to $L^2(\Omega\times\Omega)$ as soon as $d \geq 4$. From (4.21) it can be seen that $\|G\|_{L^2(X_t\times\hat{X}_s)}$ may increase when the sets X_t, $\hat{X}_s$ approach each other. The construction of $\hat{X}_s$, however, ensures

$$\sigma := \operatorname{dist}(X_t,\hat{X}_s) \geq \frac{1}{2}\operatorname{dist}(X_t,X_s) \geq \frac{1}{2\eta}\operatorname{diam} X_t$$

as well as $2\eta\sigma \geq \operatorname{diam} X_s$ due to (4.17). Hence (4.21) implies for the case $d \geq 3$

$$\|G\|_{L^2(X_t\times\hat{X}_s)} \leq \frac{c_d(\Lambda_{\mathscr{L}}/\lambda_{\mathscr{L}},\beta,\gamma,\delta)}{\lambda_{\mathscr{L}}}\sigma^{2-d}\sqrt{(\operatorname{vol} X_t)(\operatorname{vol}\hat{X}_s)}.$$

Using $\operatorname{vol}\hat{X}_s \leq \omega_d(\frac{1}{2}\operatorname{diam}\hat{X}_s)^d \leq \omega_d(\eta+1/2)^d\sigma^d$ and $\operatorname{vol} X_t \leq \omega_d(\eta\sigma)^d$, where ω_d is the volume of the unit ball in $\mathbb{R}^d$, we see that

$$\|G\|_{L^2(X_t\times\hat{X}_s)} \leq \bar{c}_\eta \frac{c_d(\Lambda_{\mathscr{L}}/\lambda_{\mathscr{L}},\beta,\gamma,\delta)}{\lambda_{\mathscr{L}}}\sigma^2 \quad \text{with } \bar{c}_\eta := \omega_d(\eta(\eta+1/2))^{d/2}.$$

Let t^* and s^* be the fathers of t and s, respectively. Then

$$\eta\operatorname{dist}(X_{t^*},X_{s^*}) \leq \max\{\operatorname{diam} X_{t^*},\operatorname{diam} X_{s^*}\}$$

and it follows from (1.22) that

$$\begin{aligned}\operatorname{dist}(X_t,\hat{X}_s) &\leq \operatorname{dist}(X_t,X_s) \leq \operatorname{dist}(X_{t^*},X_{s^*}) + \operatorname{diam} X_{t^*} + \operatorname{diam} X_{s^*}\\ &\leq (\eta^{-1}+2)\max\{\operatorname{diam} X_{t^*},\operatorname{diam} X_{s^*}\} \leq (\eta^{-1}+2)c_g^{1/d}2^{-\ell/d},\end{aligned}$$

where ℓ denotes the level of t in T_I. We obtain

$$\|(B-B_{\mathscr{H}})_b\|_2 \leq \frac{\hat{c}_d(\Lambda_{\mathscr{L}}/\lambda_{\mathscr{L}},\beta,\gamma,\delta)}{\lambda_{\mathscr{L}}}\varepsilon 4^{-\ell/d}.$$

From the proof of Theorem 2.16 we see that

$$\begin{aligned}\|B-B_{\mathscr{H}}\|_2 &\leq c_{\mathrm{sp}}\sum_{\ell=0}^{L(T_I)}\|(B-B_{\mathscr{H}})_b\|_2 \leq c_{\mathrm{sp}}\hat{c}_d(\Lambda_{\mathscr{L}}/\lambda_{\mathscr{L}},\beta,\gamma,\delta)\frac{\varepsilon}{\lambda_{\mathscr{L}}}\sum_{\ell=0}^{L(T_I)}4^{-\ell/d}\\ &\leq c_{\mathrm{sp}}\frac{\hat{c}_d(\Lambda_{\mathscr{L}}/\lambda_{\mathscr{L}},\beta,\gamma,\delta)}{1-4^{-1/d}}\frac{\varepsilon}{\lambda_{\mathscr{L}}}\end{aligned}$$

yields (4.30). Using (4.22), the case $d=2$ can be treated in a similar way. □

Remark 4.26. Assume that each (possibly non-convex) set X_t has a convex superset Y_t satisfying the admissibility condition (4.11). Then Theorem 4.25 remains valid for $X_t\times X_s$. Since the admissibility condition is checked for such supersets, i.e., cubes or spheres, the assumption on the convexity of X_t in Theorem 4.25 is reasonable even for practical purposes; see Example 1.13.

The previous theorem shows that we are able to approximate the discrete inverse of $\mathscr{L}$ by $\mathscr{H}$-matrices. Our aim, however, is to prove that the inverse of the stiffness matrix A possesses this property. For this purpose we use the fact that A^{-1} can be approximated by $M^{-1}BM^{-1}$. The last product, in turn, can be approximated by $\mathscr{H}$-matrices.

4.1.5.1 Relating A^{-1} and B

The finite element approximation is related with the Ritz projection

$$P_h = \mathscr{J} A^{-1} \mathscr{J}^* \mathscr{L} : H_0^1(\Omega) \to V_h,$$

which maps the solution $u \in H_0^1(\Omega)$ of the variational problem (4.5) to the finite element solution $u_h = P_h u$ of

$$a(u_h, v_h) = l(v_h) \quad \text{for all } v_h \in V_h.$$

The FE error is then given by $e_h(u) := \|u - P_h u\|_{L^2(\Omega)}$ and the weakest form of the finite element convergence is described by

$$e_h(u) \le \varepsilon_h \|f\|_{H^{-1}(\Omega)} \quad \text{for all } u = \mathscr{L}^{-1} f,\ f \in H^{-1}(\Omega), \tag{4.31}$$

where $\varepsilon_h \to 0$ as $h \to 0$. Due to our quite weak assumptions on the smoothness of the coefficients in (4.2), one cannot specify the convergence behavior of ε_h for $h \to 0$.

In the proof of the following lemma we will require the L^2-orthoprojection $Q_h : L^2(\Omega) \to V_h$ defined by

$$Q_h := \mathscr{J} M^{-1} \mathscr{J}^*.$$

Lemma 4.27. *Let $c_{\mathscr{J},1}$, $c_{\mathscr{J},2}$ and ε_h be the quantities from* (4.16) *and* (4.31)*. Then*

$$\|A^{-1} - M^{-1} B M^{-1}\|_2 \le \frac{c^2_{\mathscr{J},2}}{c^4_{\mathscr{J},1}} \varepsilon_h.$$

Proof. Let $x, y \in \mathbb{R}^d$ and $f_h = \mathscr{J} x$, $v_h = \mathscr{J} y \in V_h$. Then, using $B = \mathscr{J}^* \mathscr{L}^{-1} \mathscr{J}$ and the projections from above, we have

$$\begin{aligned}
((MA^{-1}M - B)x, y)_h &= ((MA^{-1}M - \mathscr{J}^* \mathscr{L}^{-1} \mathscr{J}) M^{-1} \mathscr{J}^* f_h, M^{-1} \mathscr{J}^* v_h)_h \\
&= \big((\mathscr{J} A^{-1} \mathscr{J}^* - \mathscr{J} M^{-1} \mathscr{J}^* \mathscr{L}^{-1} \mathscr{J} M^{-1} \mathscr{J}^*) f_h, v_h \big)_{L^2(\Omega)} \\
&= (P_h \mathscr{L}^{-1} f_h - Q_h \mathscr{L}^{-1} Q_h f_h, v_h)_{L^2(\Omega)} \\
&= (Q_h (P_h - \mathscr{I}) \mathscr{L}^{-1} f_h, v_h)_{L^2(\Omega)} \\
&\le e_h(\mathscr{L}^{-1} f_h) \|v_h\|_{L^2(\Omega)} \le \varepsilon_h \|f_h\|_{L^2(\Omega)} \|v_h\|_{L^2(\Omega)} \\
&\le c^2_{\mathscr{J},2} \varepsilon_h \|x\|_h \|y\|_h,
\end{aligned}$$

which proves

$$\begin{aligned}\|A^{-1} - M^{-1}BM^{-1}\|_2 &= \|M^{-1}(MA^{-1}M - B)M^{-1}\|_2 \\ &\le c_{\mathscr{J},1}^{-4}\|MA^{-1}M - B\|_2 \le c_{\mathscr{J},1}^{-4}c_{\mathscr{J},2}^2\varepsilon_h\end{aligned}$$

and hence the assertion. □

In Sect. 4.1.2 we have shown that B and M^{-1} can be approximated by $\mathscr{H}$-matrices $B_{\mathscr{H}} \in \mathscr{H}(T_{I\times I}, k_B)$ and $N_{\mathscr{H}} \in \mathscr{H}(T_{I\times I}, k_N)$, respectively. Therefore, the natural approach is to use

$$C_{\mathscr{H}} := N_{\mathscr{H}}B_{\mathscr{H}}N_{\mathscr{H}}$$

as an approximant of A^{-1}. We may assume that $k_B \ge k_N$. Due to Theorem 2.30, the exact product $C_{\mathscr{H}}$ belongs to $\mathscr{H}(T_{I\times I}, k)$ provided that $k \ge k_C := ck_BL^2(T_I)$. The estimation of the spectral norm of

$$\begin{aligned}M^{-1}BM^{-1} - N_{\mathscr{H}}B_{\mathscr{H}}N_{\mathscr{H}} &= (M^{-1} - N_{\mathscr{H}})BM^{-1} + N_{\mathscr{H}}(B - B_{\mathscr{H}})M^{-1} \\ &\quad + N_{\mathscr{H}}B_{\mathscr{H}}(M^{-1} - N_{\mathscr{H}})\end{aligned}$$

by

$$\|M^{-1} - N_{\mathscr{H}}\|_2(\|B\|_2\|M^{-1}\|_2 + \|N_{\mathscr{H}}\|_2\|B_{\mathscr{H}}\|_2) + \|N_{\mathscr{H}}\|_2\|M^{-1}\|_2\|B - B_{\mathscr{H}}\|_2$$

is obvious. Let $\varepsilon_N := \|M^{-1} - N_{\mathscr{H}}\|_2$, $\varepsilon_B := \|B - B_{\mathscr{H}}\|_2$. Since $\varepsilon_N \le \|M^{-1}\|_2$, $\varepsilon_B \le \|B\|_2$ and since due to (4.16) it holds that $\|B\|_2, \|M^{-1}\|_2 = \mathscr{O}(1)$, we obtain

$$\|M^{-1}BM^{-1} - N_{\mathscr{H}}B_{\mathscr{H}}N_{\mathscr{H}}\|_2 \le c_1(\varepsilon_N + \varepsilon_B). \tag{4.32}$$

The combination of Lemma 4.27 and (4.32) yields

$$\|A^{-1} - N_{\mathscr{H}}B_{\mathscr{H}}N_{\mathscr{H}}\|_2 \le c_1(\varepsilon_N + \varepsilon_B) + c_2\varepsilon_h,$$

where $c_2 := c_{\mathscr{J},1}^{-4}c_{\mathscr{J},2}^2$. In order to guarantee that

$$\max\{\varepsilon_N, \varepsilon_B\} \le \frac{c_2}{c_1}\varepsilon_h, \tag{4.33}$$

according to Theorem 4.25 and Theorem 4.12 the ranks k_B and k_N have to be chosen of the following orders

$$k_B \sim |\log\varepsilon_h|^{d+1} \quad \text{and} \quad k_N \sim |\log\varepsilon_h|^d.$$

With the choices (4.33) we obtain

$$\|A^{-1} - N_{\mathscr{H}}B_{\mathscr{H}}N_{\mathscr{H}}\|_2 \le 3c_2\varepsilon_h, \tag{4.34}$$

which shows that the already existing finite element error $c_2\varepsilon_h$ is only slightly increased by the $\mathscr{H}$-matrix approximation. The resulting rank for $C_{\mathscr{H}} = N_{\mathscr{H}}B_{\mathscr{H}}N_{\mathscr{H}}$

is bounded by $k_C = ck_BL^2(T_I)$. Thus, $C_{\mathscr{H}}$ approximates A^{-1} as described in (4.34) and belongs to $\mathscr{H}(T_{I\times I}, k)$ for all $k \geq ck_BL^2(T_I)$.

Theorem 4.28. *Let $\varepsilon_h > 0$ be defined in* (4.31) *and let $L = L(T_{I\times I}) \sim \log n$ denote the depth of $T_{I\times I}$. Then there is an $\mathscr{H}$-matrix $C_{\mathscr{H}} \in \mathscr{H}(T_{I\times I}, k_C)$, $k_C \sim L^2|\log\varepsilon_h|^{d+1}$, such that*

$$\|A^{-1} - C_{\mathscr{H}}\|_2 \leq \varepsilon_h. \tag{4.35}$$

If $\varepsilon_h \sim h^\beta$ with some $\beta > 0$, then $k_C \sim \log^{d+3} n$ holds.

Proof. Since $h^{-1} \sim n^{1/d}$, the asymptotic behavior of $|\log\varepsilon_h|$ is $\log n$. □

Since $\lambda_{\mathscr{L}}$ in (4.3) is of size $\mathscr{O}(1)$ (without loss of generality, we may scale the problem so that $\lambda_{\mathscr{L}} = 1$), also $\|A^{-1}\|_2 = \mathscr{O}(1)$ holds. Hence, the absolute error (4.35) may be changed into a relative one: $\|A^{-1} - C_{\mathscr{H}}\|_2 \leq \varepsilon_h\|A^{-1}\|_2$.

4.1.6 Numerical Experiments

In this section the practical influence of the various terms of the differential operator (4.2) on the efficiency and accuracy of the $\mathscr{H}$-matrix inverse is investigated. For simplicity all tests are performed on a uniform triangulation of the unit square $\Omega := (0,1)\times(0,1)$ in $\mathbb{R}^2$. In each case the stiffness matrix A is built in the $\mathscr{H}$-matrix format. Then the inversion algorithm from Sect. 2.8 is applied to it with a relative rounding precision $\varepsilon_{\mathscr{H}}$. Hence, the blockwise rank is adaptively chosen and is therefore expected to vary among the blocks. The maximum rank among all blocks in P will be denoted by $k_{\max}$. All tests were carried out on an Athlon64 (2 GHz) workstation with 12 GB of core memory. The minimal block size $n_{\min}$ was chosen 128.

Let $u_h := \mathscr{J}x \in V_h$ be the FE solution; i.e., $x \in \mathbb{R}^n$ is the solution of $Ax = b$ and b is the vector with the components

$$b_i = \int_\Omega f\varphi_i \,\mathrm{d}x, \quad i = 1,\ldots,n.$$

Furthermore, let $\tilde{u}_h = \mathscr{J}\tilde{x}$, where $\tilde{x} := C_{\mathscr{H}}b$ and $C_{\mathscr{H}}$ is the computed $\mathscr{H}$-matrix approximant of A^{-1}. Since

$$\|x - \tilde{x}\|_2 = \|x - C_{\mathscr{H}}Ax\|_2 \leq \|I - C_{\mathscr{H}}A\|_2\,\|x\|_2, \tag{4.36}$$

the expression $\|I - C_{\mathscr{H}}A\|_2$ is an upper bound on the relative accuracy of $\tilde{x}$ compared with the solution x. Hence, in the following computations we will rely on the expression $\mathscr{E} := \|I - C_{\mathscr{H}}A\|_2$ as a measure of accuracy of $\tilde{u}_h$. Note that $\tilde{u}_h$ cannot be a better approximation of u than u_h is, because the proposed method is built on top of the finite element method.

4.1.6.1 Principal Parts

In the first example we consider operators $\mathscr{L} = -\mathrm{div}\, C\nabla$ with coefficients C of the form

$$C(x) = \begin{bmatrix} 1 & 0 \\ 0 & \alpha(x) \end{bmatrix}, \quad x \in \Omega, \tag{4.37}$$

where $\alpha(x) = 1$ in the lower region of Fig. 4.2 and a random number from the interval $[0,a]$ in the remaining part of the unit square. The amplitude a will be used

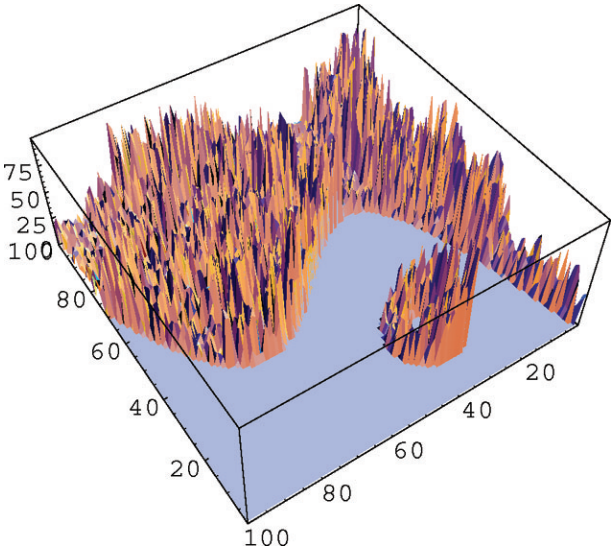


Fig. 4.2 The coefficient $\alpha(x)$.

to demonstrate that the $\mathscr{H}$-matrix inverse is not sensitive to the size of the non-smooth coefficients. In order to avoid averaging effects, the coefficient α possesses a two-level random structure: the randomly chosen coefficient on each triangle is multiplied by a coefficient chosen randomly on a length scale $\sqrt{h}$, where the grid size h is defined by $h(\sqrt{n/2}+1) = 1$. Since the arising finite element stiffness matrices are symmetric, we make use of an obvious symmetric variant of the inversion procedure from Sect. 2.8. The symmetric inversion saves about half of the computational costs.

In the following tables the accuracy $\mathscr{E} = \|I - C_{\mathscr{H}}A\|_2$ of the $\mathscr{H}$-matrix $C_{\mathscr{H}}$, the CPU time consumption, and the storage requirement are compared for different amplitudes a and different problem sizes n. In Table 4.1 we fix the rounding precision $\varepsilon_{\mathscr{H}} = 1_{10}-6$ and compare the asymptotic behavior of the results in Fig. 4.3 with functions $n\log^* n$. Note that in Theorem 4.28 we have proved that k is of the order $\log^2 n$ if $\varepsilon_{\mathscr{H}}$ is assumed to be constant. Therefore, Theorem 2.6 implies a theoretical bound of $n\log^3 n$ on the storage complexity, while Theorem 2.33 leads to a bound

Table 4.1 $\mathscr{H}$-matrix inversion with fixed accuracy.

	$a=1$				$a=100$			
n	time	$\mathscr{E}$	$k_{\max}$	MB	time	$\mathscr{E}$	$k_{\max}$	MB
14 400	6s	$7.5_{10}-4$	22	39	6s	$1.7_{10}-3$	22	39
38 025	27s	$3.0_{10}-3$	24	123	27s	$5.8_{10}-3$	24	123
65 025	56s	$4.3_{10}-3$	24	247	57s	$1.1_{10}-2$	24	247
129 600	160s	$1.0_{10}-2$	30	564	156s	$1.9_{10}-2$	30	544
278 784	418s	$2.4_{10}-2$	25	1 339	416s	$4.7_{10}-2$	25	1 321
529 984	1 129s	$4.7_{10}-2$	30	2 864	1 134s	$1.2_{10}-1$	30	2 800

$n\log^6 n$ on the number of operations. Apparently, the CPU time scales like $n\log^5 n$ and the storage requirement is of the order $n\log^2 n$ in practice. Compared with the standard inversion procedure, which has complexity n^3, the $\mathscr{H}$-matrix inverse thus scales almost linearly. The $\mathscr{H}$-matrix approximant of the inverse for $n = 529\,984$ unknowns requires at most 3 GB of storage while the exact inverse would need 2 093 GB.

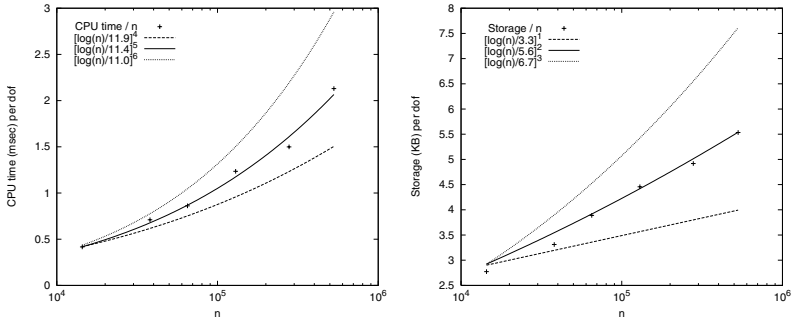


Fig. 4.3 CPU time and storage compared with $n\log^5 n$ and $n\log^2 n$.

In Table 4.2 the truncation accuracy $\varepsilon_{\mathscr{H}}$ is chosen such that $\mathscr{E}$ is of order h. Since according to Theorem 4.28 we have that k is of the order at most $\log^2 n|\log\varepsilon|^{d+1} \sim \log^5 n$, we expect additional logarithmic factors in the asymptotic complexities. Figure 4.4 shows that instead of the estimated complexities $n\log^{12} n$ and $n\log^7 n$ asymptotic complexities of the order $n\log^7 n$ and $n\log^3 n$ can be observed in practice.

In the next set of tests the same quantities for a smooth but oscillating coefficient α in the principal part are computed; i.e., α is chosen to be the function

$$\alpha(x) = a[1+\cos(2\pi bx)\sin(2\pi by)].$$

Table 4.2 $\mathscr{H}$-matrix inversion with $\mathscr{E} \sim h$.

		$a=1$				$a=100$			
n	$\varepsilon_{\mathscr{H}}$	time	$\mathscr{E}$	$k_{\max}$	MB	time	$\mathscr{E}$	$k_{\max}$	MB
14 400	$5_{10}-6$	6s	$3.5_{10}-3$	22	36	6s	$1.4_{10}-2$	21	36
38 025	$2_{10}-6$	25s	$5.6_{10}-3$	22	118	25s	$1.2_{10}-2$	21	118
65 025	$5_{10}-7$	59s	$2.5_{10}-3$	24	256	60s	$5.8_{10}-3$	24	256
129 600	$2_{10}-7$	186s	$2.2_{10}-3$	31	618	183s	$4.8_{10}-3$	31	597
278 784	$5_{10}-8$	596s	$1.4_{10}-3$	32	1 602	554s	$3.2_{10}-3$	30	1 574
529 984	$2_{10}-8$	1 685s	$1.1_{10}-3$	32	3 638	1 667s	$2.3_{10}-3$	33	3 539

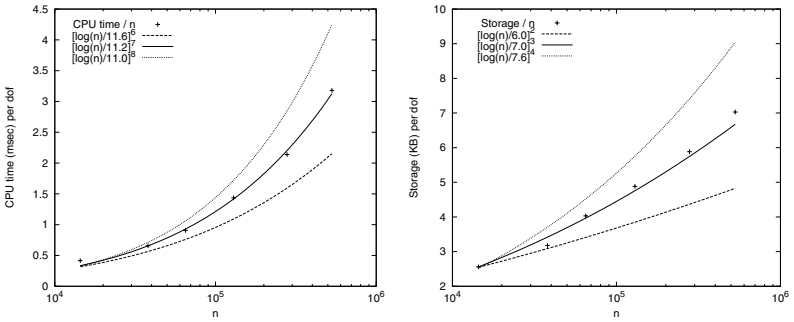


Fig. 4.4 CPU time and storage compared with $n\log^7 n$ and $n\log^3 n$.

By changing the coefficient a we are able to prescribe the amplitude of the oscillation and by b the number of oscillations in x- and y-direction of $\Omega = (0,1)\times(0,1)$. Table 4.3 contains the results for $n = 65\,025$ and $\varepsilon_{\mathscr{H}} = 5_{10}-7$.

Table 4.3 $\mathscr{H}$-matrix for operators with oscillating coefficients.

	$a=1$				$a=100$			
b	time	$\mathscr{E}$	$k_{\max}$	MB	time	$\mathscr{E}$	$k_{\max}$	MB
1	64s	$3.0_{10}-3$	30	269	93s	$7.6_{10}-3$	40	288
10	61s	$2.9_{10}-3$	25	263	87s	$2.2_{10}-3$	36	283
100	61s	$2.9_{10}-3$	25	263	94s	$2.2_{10}-3$	37	287
1000	62s	$2.7_{10}-3$	25	262	94s	$1.5_{10}-3$	37	287

The previous experiments show that at least in the absence of lower-order terms the accuracy, the CPU time, and the storage needed to compute the approximant do only slightly depend on the size of the coefficients. Especially, nonsmooth coefficients do not affect the computational complexity.

4.1.6.2 Convection-Diffusion

Next, operators of the type

$$\mathscr{L} = -\Delta + \beta \cdot \nabla$$

will be considered. In the first example the convection coefficient β is randomly chosen; i.e., $\beta(x) \in [-a,a]^2$ for $x \in \Omega$. Table 4.4 shows $\mathscr{E} = \|I - C_{\mathscr{H}} A\|_2$ for different parameters a. As a second example we investigate operators

Table 4.4 $\mathscr{H}$-matrix inversion for convection-diffusion equations.

		$a = 10$				$a = 100$			
n	$\varepsilon_{\mathscr{H}}$	time	$\mathscr{E}$	$k_{\max}$	MB	time	$\mathscr{E}$	$k_{\max}$	MB
14 641	$5_{10}-6$	13s	$3.7_{10}-3$	21	73	13s	$4.6_{10}-3$	21	72
38 416	$2_{10}-6$	60s	$5.6_{10}-3$	19	235	60s	$5.6_{10}-3$	20	234
65 025	$5_{10}-7$	160s	$2.6_{10}-3$	23	491	160s	$2.5_{10}-3$	23	490
129 600	$2_{10}-7$	468s	$2.1_{10}-3$	20	1 185	472s	$2.3_{10}-3$	21	1 183
278 784	$5_{10}-8$	1 590s	$1.2_{10}-3$	25	3 171	1 595s	$1.2_{10}-3$	25	3 172

$$\mathscr{L} u = -\varepsilon \Delta u + u_x + u_y$$

for different parameters $\varepsilon > 0$. In the following tests (see Table 4.5) we restrict ourselves to moderately sized ε. The rounding precisions $\varepsilon_{\mathscr{H}}$ were chosen as in the last table.

Table 4.5 $\mathscr{H}$-matrix inversion for dominating convection.

	$\varepsilon = 0.1$			$\varepsilon = 0.01$			$\varepsilon = 0.001$		
n	time	$\mathscr{E}$	MB	time	$\mathscr{E}$	MB	time	$\mathscr{E}$	MB
14 641	12s	$2.7_{10}-3$	75	13s	$2.3_{10}-4$	80	37s	$8.9_{10}-5$	132
38 416	59s	$3.6_{10}-3$	242	69s	$3.6_{10}-4$	273	155s	$7.6_{10}-5$	411
65 025	132s	$1.6_{10}-3$	516	159s	$1.6_{10}-4$	576	259s	$2.3_{10}-5$	707
129 600	426s	$1.3_{10}-3$	1 227	506s	$1.8_{10}-4$	1 372	581s	$1.5_{10}-5$	1 371
278 784	1 336s	$7.9_{10}-4$	3 160	1 630s	$1.1_{10}-4$	3 532	1 322s	$7.0_{10}-6$	2 835

Since the above results show that it is possible to find an $\mathscr{H}$-matrix $C_{\mathscr{H}}$ which approximates A^{-1}, from (4.36) it is obvious that $\tilde{u}_h$ approximates the finite element solution u_h. This is especially true in the presence of boundary layers as illustrated in the following example. The solution of

$$-\varepsilon \Delta u + u_x + u_y = f,$$

where

$$f(x,y) = (x+y)\left(1 - e^{(x-1)/\varepsilon} e^{(y-1)/\varepsilon}\right) + (x-y)\left(e^{(y-1)/\varepsilon} - e^{(x-1)/\varepsilon}\right)$$

with zero boundary conditions is known to be

$$u(x,y) = xy\left(1 - e^{(x-1)/\varepsilon}\right)\left(1 - e^{(y-1)/\varepsilon}\right).$$

Figure 4.5 compares the restrictions of u and $\tilde{u}_h$ to the set $\{(x,x), x \in (0,1)\}$ for $\varepsilon = 0.01$ and $n = 14\,641$. Apparently, the proposed inversion procedure is able to

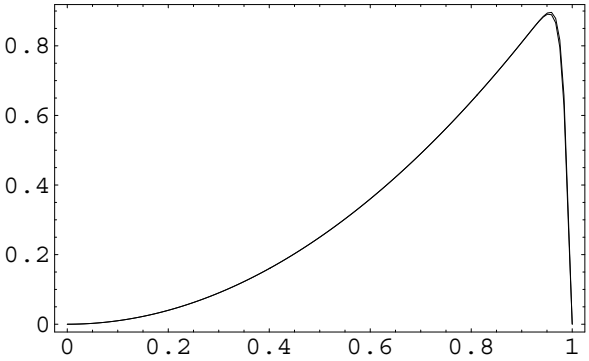


Fig. 4.5 The solutions u and $\tilde{u}_h$.

handle boundary layers as long as the underlying finite element method is stable.

In the next example (see Table 4.6) we consider convection in a direction that is aligned with the grid; i.e., operators

$$\mathscr{L}u = -\varepsilon\Delta u + u_x$$

for different parameters $\varepsilon > 0$ are investigated.

Table 4.6 $\mathscr{H}$-matrix inversion for aligned convection.

	$\varepsilon = 0.1$			$\varepsilon = 0.01$			$\varepsilon = 0.001$		
n	time	$\mathscr{E}$	MB	time	$\mathscr{E}$	MB	time	$\mathscr{E}$	MB
14 641	13s	$2.9_{10}-3$	74	14s	$3.6_{10}-4$	82	28s	$8.3_{10}-5$	112
38 416	60s	$4.2_{10}-4$	239	73s	$5.2_{10}-4$	271	144s	$6.3_{10}-5$	372
65 025	135s	$1.9_{10}-3$	512	162s	$1.5_{10}-4$	566	275s	$1.9_{10}-5$	703
129 600	429s	$1.3_{10}-3$	1 205	498s	$1.6_{10}-4$	1 327	735s	$1.5_{10}-5$	1 503
278 784	1 390s	$9.3_{10}-4$	3 134	1 676s	$1.3_{10}-4$	3 447	2 250s	$6.6_{10}-6$	3 778

It is well known that if the ratio ε/h becomes small, the finite element discretization suffers from the loss of stability. As a consequence, the stiffness matrix becomes ill-conditioned. It may even happen that A is not invertible. This behavior is observable in Tables 4.5 and 4.6 for small n. When changing $\varepsilon = 0.01$ to $\varepsilon = 0.001$, the

CPU time for $n = 14\,641$ increases by a factor of 2, while it is less influenced in the case $n = 278\,784$, where the discretization is stable.

4.1.6.3 Diffusion-Reaction

As a third kind of example we consider operators $\mathscr{L} = -\Delta + \delta$. Table 4.7 shows the approximation results if the reaction term $\delta(x) \in [0,a]$ is randomly chosen for $x \in \Omega$. By adding a positive δ to the operator $-\Delta$, the distance of the spectrum to

Table 4.7 $\mathscr{H}$-matrix inversion for positive reaction terms.

	$a = 10$			$a = 100$			$a = 1000$		
n	time	$\mathscr{E}$	MB	time	$\mathscr{E}$	MB	time	$\mathscr{E}$	MB
14 641	6s	$2.3_{10}-3$	37	6s	$2.7_{10}-4$	38	7s	$2.6_{10}-5$	41
38 416	26s	$3.6_{10}-3$	119	27s	$5.6_{10}-4$	124	30s	$2.9_{10}-5$	134
65 025	65s	$1.6_{10}-3$	250	69s	$3.3_{10}-4$	260	77s	$2.3_{10}-5$	280
129 600	190s	$1.3_{10}-3$	598	197s	$3.1_{10}-4$	624	219s	$2.5_{10}-5$	663
278 784	635s	$8.0_{10}-4$	1 601	628s	$1.7_{10}-4$	1 653	695s	$1.4_{10}-5$	1 753
597 529	1 831s	$6.5_{10}-4$	3 982	1 899s	$1.3_{10}-4$	4 115	2 128s	$1.1_{10}-5$	4 377

the origin is increased. Hence, the larger δ, the better the approximation works. We used the mentioned symmetric inversion procedure.

Negative δ, i.e., the Helmholtz operator, can also be handled as long as the inverse of $\mathscr{L}$ is guaranteed to exist. Each column of Table 4.8 shows the approximation results for the respective δ in the case $n = 129\,600$. In order to be able to guarantee $\|I - C_{\mathscr{H}}A\|_2 \sim h$, we have to choose a higher truncation accuracy $\varepsilon_{\mathscr{H}}$ if the modulus of δ is increased. Note that δ is now a constant. For large wave numbers

Table 4.8 $\mathscr{H}$-inverse for the Helmholtz operator.

δ	-1	-10	-100	$-1\,000$	$-10\,000$
$\varepsilon_{\mathscr{H}}$	$2.0_{10}-07$	$1.0_{10}-07$	$1.0_{10}-09$	$3.0_{10}-10$	$3.0_{10}-12$
$\mathscr{E}$	$2.2_{10}-03$	$2.5_{10}-03$	$2.3_{10}-03$	$2.2_{10}-03$	$2.6_{10}-03$
time	187s	198s	283s	347s	840s
$k_{\max}$	20	20	24	25	42
MB	592	607	739	908	1 551

the inversion procedure can be applied, but becomes less efficient.

From the numerical experiments above we conclude that the $\mathscr{H}$-matrix inversion is robust with respect to nonsmooth and anisotropic coefficients. Even convection-diffusion problems with dominating convection can be solved efficiently without special adaptation of the algorithm to this class of problems. Hence, the inversion procedure can be applied whenever a stable discretization of the operator $\mathscr{L}$ is available. In the case of singularly perturbed problems it is shown in [175] that in order

to be able to apply low-rank approximations, one has to reorder to indices appropriately.

Although we have given evidence that an approximation of A^{-1} can be computed in the set of $\mathscr{H}$-matrices with logarithmic-linear complexity, the absolute costs of the inversion procedure can and ought to be improved. Especially for three-dimensional applications, the approximation of the factors of the LU decomposition of A will turn out to be more efficient. Therefore, we delay three-dimensional examples to the numerical experiments of the approximate LU decomposition in Sect. 4.4.

4.2 Schur Complements

For domain decomposition methods (see, for instance, [238]) the efficient treatment of Schur complements is of particular importance. Efficient numerical methods for the data-sparse approximation of the elliptic **Poincaré-Steklov operators** (PSO) are considered in [160]; see [161, 157] for data-sparse methods for PSOs in the case of refined meshes. Hierarchical matrices are used in [135] for the approximation of Schur complements on the interface. In this section it will be shown that Schur complements of sub-blocks of FE stiffness matrices A can be approximated by $\mathscr{H}$-matrices. This result will lay ground to the approximation of the factors L and U arising from the LU decomposition of A in the next Sect. 4.3.

Assume that the Galerkin stiffness matrix $A \in \mathbb{R}^{n\times n}$ is partitioned in the following way:

$$A = \begin{bmatrix} A_{rr} & A_{rr'} \\ A_{r'r} & A_{r'r'} \end{bmatrix}, \tag{4.38}$$

where $r \subset I = \{1,\dots,n\}$ and $r' := I \setminus r$. We will show that the usually fully populated Schur complement

$$S := A_{r'r'} - A_{r'r} A_{rr}^{-1} A_{rr'}$$

of A_{rr} in A can be approximated by an $\mathscr{H}$-matrix with blockwise rank k, where k depends only logarithmically on both the approximation accuracy and n. For this purpose it is crucial to notice that in the case of Dirichlet problems A_{rr} in (4.38) is nothing but the Galerkin matrix of $\mathscr{L}$ if we replace Ω by the subdomain X_r. Hence, Theorem 4.28 guarantees that an $\mathscr{H}$-matrix approximant exists for A_{rr}^{-1}. Additionally, we assume that there is a constant $c > 0$ such that

$$\|A_{rr}^{-1}\|_2 \le c\|A^{-1}\|_2. \tag{4.39}$$

The previous estimate holds, for instance, if A is symmetric positive definite, because setting $\hat{x} := [x^T, 0]^T \in \mathbb{R}^I$ for $x \in \mathbb{R}^r$ we have

$$\|A_{rr}^{-1}\|_2 = \sup_{x\in\mathbb{R}^r} \frac{x^T x}{x^T A_{rr} x} = \sup_{\hat{x}\in\mathbb{R}^I} \frac{\hat{x}^T \hat{x}}{\hat{x}^T A \hat{x}} \le \|A^{-1}\|_2.$$

Remark 4.29. Since the proof of Theorem 4.28 is based on the FE error estimate (4.31), we were only able to show existence of approximants with an accuracy which is of the order of the FE error ε_h. This is not a restriction since a higher accuracy in the approximation of the inverse would be superposed by the FE error in the solution anyhow. However, the numerical experiments from the previous section show that the above result is true for any accuracy. In addition, the algebraic approach from Sect. 4.1.3 allows to prove arbitrary precision approximations. Therefore, in this section we assume that for any $\varepsilon > 0$ there is $C_{\mathscr{H}} \in \mathscr{H}(T_{I\times I}, k)$ with $k := |\log \varepsilon|^{d+1}(\log n)^2$ such that

$$\|A^{-1} - C_{\mathscr{H}}\|_2 < c\,\varepsilon \|A^{-1}\|_2, \tag{4.41}$$

where $c > 0$ depends on the size of the coefficients of $\mathscr{L}$, the diameter of Ω and the cluster parameter η.

Let $t \in T_I$ be a cluster. By

$$\mathscr{N}(t) := \{i \in I : \operatorname{dist}(X_i, X_t) = 0\}$$

we denote a neighborhood of t and

$$\mathscr{F}_\eta(t) := \{i \in I : \eta \operatorname{dist}(X_i, X_t) > \operatorname{diam} X_t\}$$

will be referred to as the far-field of t. Furthermore, we define the ratios

$$q := \frac{\max_{i\in I} \operatorname{diam} X_i}{\min_{t\in T_I} \operatorname{diam} X_t} \quad \text{and} \quad \bar{q} := \max_{t\in T_I} \frac{\operatorname{diam} X_{\mathscr{N}(t)}}{\operatorname{diam} X_t} \le 1 + 2q.$$

Since the minimal cluster size $n_{\min}$ is usually chosen larger than 20, realistic values for $\bar{q}$ can be expected to be close to 1. The size of q depends on the uniformity of $\{X_i\}_{i\in I}$.

We need the following basic lemma which states that the neighborhood of t is in the far-field of the neighborhood of s if t is in the far-field of s.

Lemma 4.30. *Let* $0 < \eta < (q+\bar{q})^{-1}$. *If* $t \subset \mathscr{F}_\eta(s)$, *then*

$$\mathscr{N}(t) \subset \mathscr{F}_{\tilde{\eta}}(\mathscr{N}(s)), \quad \textit{where} \quad \tilde{\eta} = \frac{\bar{q}\eta}{1-(q+\bar{q})\eta}.$$

Proof. Since $\max_{i\in I} \operatorname{diam} X_i \le q \operatorname{diam} X_s$, we obtain for $x \in X_{\mathscr{N}(t)}$ and $y \in X_{\mathscr{N}(s)}$ that

$$\begin{aligned} |x-y| &\ge \operatorname{dist}(X_t, X_s) - \max_{i\in I} \operatorname{diam} X_i - \operatorname{diam} X_{\mathscr{N}(s)} \\ &> (\eta^{-1} - q)\operatorname{diam} X_s - \operatorname{diam} X_{\mathscr{N}(s)} \\ &\ge \left[\frac{1}{\bar{q}}(\eta^{-1} - q) - 1\right] \operatorname{diam} X_{\mathscr{N}(s)}, \end{aligned}$$

which proves the assertion. □

Using the previous lemma we can now prove that the Schur complement S of FE Galerkin matrices A can be approximated in the desired way.

Theorem 4.31. *Let the FE Galerkin matrix $A \in \mathbb{R}^{n\times n}$ be partitioned as in* (4.38) *with $r \neq \emptyset$. Then for the Schur complement*

$$S = A_{r'r'} - A_{r'r}A_{rr}^{-1}A_{rr'}$$

of A_{rr} in A and all $\varepsilon > 0$ there is $S_{\mathscr{H}} \in \mathscr{H}(T_{I\times I}, k_S)$, where $k_S \sim |\log\varepsilon|^{d+1}(\log|r|)^2$, such that

$$\|S - S_{\mathscr{H}}\|_2 < \varepsilon L(T_I)\,\mathrm{cond}_2(A)\|A\|_2. \tag{4.42}$$

Proof. We have to show that for each admissible sub-block $b = t \times s \in P$ of $r' \times r'$ and any prescribed accuracy $\varepsilon > 0$ we can find a low-rank matrix which approximates S_b with accuracy ε. Since b is admissible, $(A_{r'r'})_b = 0$ holds. Hence,

$$S_b = -A_{tr}A_{rr}^{-1}A_{rs} = -\sum_{i,j\in r} A_{ti}(A_{rr}^{-1})_{ij}A_{js}.$$

If $i \notin \mathscr{N}(t)$, then $A_{ti} = 0$. If on the other hand $j \notin \mathscr{N}(s)$, then $A_{js} = 0$. With the notation $\mathscr{N}'(t) := \mathscr{N}(t) \cap r$, we have

$$S_b = -\sum_{i\in\mathscr{N}'(t),\, j\in\mathscr{N}'(s)} A_{ti}(A_{rr}^{-1})_{ij}A_{js}.$$

Since b is admissible, $t \subset \mathscr{F}_\eta(s)$ or $s \subset \mathscr{F}_\eta(t)$ holds. According to Lemma 4.30, it follows that $\mathscr{N}(t) \subset \mathscr{F}_{\tilde\eta}(\mathscr{N}(s))$ or $\mathscr{N}(s) \subset \mathscr{F}_{\tilde\eta}(\mathscr{N}(t))$ is valid. Following (4.41) (with η replaced by $\tilde\eta$), there are $X \in \mathbb{R}^{\mathscr{N}'(t)\times k}$ and $Y \in \mathbb{R}^{\mathscr{N}'(s)\times k}$ with $k \sim |\log\varepsilon|^{d+1}(\log|r|)^2$ such that

$$\|(A_{rr}^{-1})_{\mathscr{N}'(t)\mathscr{N}'(s)} - XY^T\|_2 < \varepsilon\|A_{rr}^{-1}\|_2.$$

Let X and Y be extended to $\hat X \in \mathbb{R}^{r\times k}$ and $\hat Y \in \mathbb{R}^{r\times k}$ by adding zero rows. Observe that

$$\begin{aligned} A_{tr}\hat X\hat Y^T A_{rs} &= \sum_{i\in\mathscr{N}'(t),\, j\in\mathscr{N}'(s)} \sum_{\ell=1}^{k} A_{ti}X_{i\ell}Y_{j\ell}A_{js} \\ &= \sum_{\ell=1}^{k} \left(\sum_{i\in\mathscr{N}'(t)} A_{ti}X_{i\ell}\right)\left(\sum_{j\in\mathscr{N}'(s)} Y_{j\ell}A_{js}\right) = VW^T, \end{aligned}$$

where $V \in \mathbb{R}^{t\times k}$, $W \in \mathbb{R}^{s\times k}$ with the entries

$$V_{t\ell} := \sum_{i\in\mathscr{N}'(t)} A_{ti}X_{i\ell} \quad\text{and}\quad W_{s\ell} := \sum_{j\in\mathscr{N}'(s)} Y_{j\ell}A_{js}, \quad \ell = 1,\dots,k.$$

Define $B \in \mathbb{R}^{r\times r}$ with entries

$$B_{ij} = \begin{cases} (A_{rr}^{-1})_{ij}, & \text{if } i \in \mathcal{N}'(t) \text{ and } j \in \mathcal{N}'(s), \\ 0, & \text{else,} \end{cases}$$

then using (4.39) it follows that

$$\begin{aligned} \|S_b - VW^T\|_2 = \|A_{tr}(B - \hat{X}\hat{Y}^T)A_{rs}\|_2 &\leq \|A_{tr}\|_2 \|(A_{rr}^{-1})_{\mathcal{N}'(t)\mathcal{N}'(s)} - XY^T\|_2 \|A_{rs}\|_2 \\ &\leq \varepsilon \|A_{tr}\|_2 \|A_{rr}^{-1}\| \|A_{rs}\|_2 < c\,\varepsilon\, \mathrm{cond}_2(A)\|A\|_2. \end{aligned}$$

The assertion follows from Theorem 2.16. □

Since the spectral condition number grows polynomially with n and since the accuracy ε enters the complexity estimate only through the logarithm, we can get rid of this factor in (4.42) if the rank k_S is increased by adding a logarithmic factor.

4.3 Hierarchical *LU* Decomposition

Although efficient iterative methods are available for the solution of linear systems, the *LU* decomposition is still one of the most often used solution strategies in practice even for large-scale problems. This is mainly due to the robustness of direct solvers. In addition, factorizations have the advantage that linear systems with many right-hand sides can be solved efficiently. However, direct methods suffer from so-called **fill-in**, i.e., compared with the sparsity of A considerably more entries of the factors L and U will be nonzero. Usually, they will be fully populated up to the bandwidth of A, which for Galerkin matrices of operators (4.2) scales at least like $n^{1-1/d}$ even if the bandwidth is reduced, for instance, by the **Reverse Cuthill-McKee (RCM) algorithm** [72]. Hence, the fill-in will lead to a computational complexity of order $n^{3-2/d}$. Instead of reducing the bandwidth, the aim of the **minimum degree algorithm** [218] and **nested dissection** [99] is to reduce fill-in. When $d = 2$, the fill-in for nested dissection is of the order $n \log n$ and the complexity is $n^{3/2}$; see [100]. For $d > 2$ the complexity of nested dissection scales like $n^{3-3/d}$; cf. [187]. Despite this fact, recent multifrontal and supernodal approaches [78, 4, 230] lead to highly efficient algorithms. As a consequence, direct methods are still one of the most efficient methods if n is not too large or if $d = 2$.

The efficiency of the usual *LU* decomposition is determined by the fill-in of the factors L and U of A. Since $\mathcal{H}$-matrices are able to handle dense matrices with almost linear complexity, the fill-in is not an issue when using this structure. Due to the reordering of indices required when building binary cluster trees, we even obtain a bandwidth which is of order n. This will result in an enormous fill-in and is unavoidable as can be seen from the following example. A nested dissection approach (which leads to ternary cluster trees) to the hierarchical *LU* factorization will be presented in Sect. 4.5.

Example 4.32. A matrix entry a_{ij} in the Galerkin matrix A will in general be nonzero if the supports of the associated basis functions φ_i and φ_j have a nonempty intersec-

tion. For simplicity we investigate the situation which occurs for a regular triangulation of the unit square in $\mathbb{R}^2$. Assume that after two subdivision steps this square has been subdivided into four smaller squares of the same size each containing $n/4$ supports. The indices are reordered during the subdivision so that the kth square contains the indices $(k-1)n/4+1$ to $kn/4$, $k=1,\dots,4$. Hence, the first and the last square contain indices which differ by at least $n/2$. These squares intersect in the center of the original square. Therefore, at this point the supports of two basis functions φ_i and φ_j with $|i-j| \geq n/2$ intersect. This situation persists when the subdivision is continued since further clustering rearranges the indices only within each subsquare.

Assume that all minors of A are nonzero. Then A can be decomposed as follows

$$A = LU,$$

where L is a lower triangular and U is an upper triangular matrix. In this section it will be shown that the factors L and U can be approximated by $\mathscr{H}$-matrices $L_{\mathscr{H}}$ and $U_{\mathscr{H}}$ if any Schur complement in A has this property. From the previous Sect. 4.2 we know that this applies to FE stiffness matrices. The following proof consists of algebraic arguments only. Hence, the LU decompositions can be accelerated also for problems that do not stem from FE applications as long as the Schur complements are known to have an efficient approximant in the set of $\mathscr{H}$-matrices. The hierarchical LU decomposition differs conceptually from the **incomplete** LU **factorization** (ILU); see [222]. The ILU overcomes the problem of fill-in by setting entries in the factors L and U outside of the sparsity pattern of A to zero. Although the ILU can be equipped with a thresholding parameter, it will always result in more or less sparse factors, while the hierarchical LU decomposition is a data-sparse representation of a fully (up to the bandwidth) populated matrix approximant.

When computing pointwise LU decompositions, usually pivoting is performed in order to avoid zero or almost zero pivots. For block versions of the LU algorithm the possibilities of pivoting are limited if the blocking is given. In our case we can choose only from two possible pivots, block $t_1 \times t_1$ or block $t_2 \times t_2$, if the LU decomposition of a block $t \times t$, $t = t_1 \cup t_2$, is to be computed; see Remark 1.22. Hence, the accuracy analysis cannot rely on the advantages of pivoting. In order to show that L and U can be approximated by $\mathscr{H}$-matrices it seems natural to define the approximants

$$\tilde{L} = \begin{bmatrix} \tilde{L}_{11} & \\ \tilde{L}_{21} & \tilde{L}_{22} \end{bmatrix} \quad \text{and} \quad \tilde{U} = \begin{bmatrix} \tilde{U}_{11} & \tilde{U}_{12} \\ & \tilde{U}_{22} \end{bmatrix}$$

recursively as

$$\tilde{L}_{11}\tilde{U}_{11} = A_{11} + E_{11} \tag{4.43a}$$
$$\tilde{L}_{11}\tilde{U}_{12} = A_{12} + E_{12} \tag{4.43b}$$
$$\tilde{L}_{21}\tilde{U}_{11} = A_{21} + E_{21} \tag{4.43c}$$
$$\tilde{L}_{22}\tilde{U}_{22} = A_{22} - \tilde{L}_{21}\tilde{U}_{12} + E_{22}; \tag{4.43d}$$

replacing appropriate sub-blocks of the arising Schur complements with low-rank matrices thereby introducing the error terms E_{12} and E_{21}. The errors E_{11} and E_{22} could then be estimated by the error analysis of the block LU decomposition; see [79]. The problem with this approach is that the arising Schur complements (see (4.43d)) are not the original complements but complements that contain the perturbations from all previous approximation steps. Since it cannot be guaranteed that these perturbed complements can be approximated by $\mathscr{H}$-matrices and since their distance to the exact complements leads to unattractive estimates, we have to go a different way; cf. [23]. In the following subsection we first find a recursive relation between the Schur complement of a block b and the complements of its sub-blocks.

4.3.1 Approximating Schur Complements Hierarchically

Let $A \in \mathbb{R}^{n\times n}$ and $t,s \subset I$. With the notations $\hat{t} := \{i \in I : i \leq \max t\}$ and $\hat{s} := \{j \in I : j \leq \max s\}$ the Schur complement of the block $t \times s$ in $A_{\hat{t}\hat{s}}$ is defined as

$$S(t,s) = A_{ts} - A_{tr}A_{rr}^{-1}A_{rs}, \tag{4.44}$$

where $r := \{i \in I : i < \min t \cup s\}$; see Fig. 4.6. In the case $r = \varnothing$ this definition is

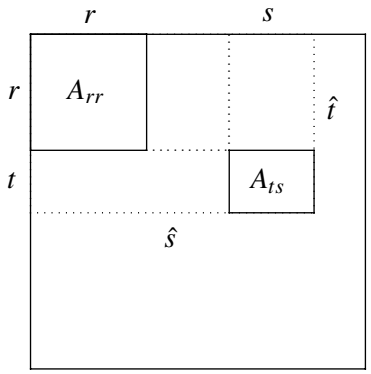


Fig. 4.6 Schur complement of a block $t \times s$.

meant to result in $S(t,s) = A_{ts}$. The expression $S(t,s)$ is the usual Schur complement for $t = s$. If $t \neq s$, then $S(t,s)$ is the restriction of the usual Schur complement of A_{rr} to $t \times s$. Note that if t_1, t_2 are the sons of t, then $S(t_2,t_2)$ does not coincide with the sub-block in the rows t_2 and columns t_2 of $S(t,t)$ in general. The following lemma will show the right relation between the complement of a block and the complements of its sub-blocks. For the ease of notation we first consider the case of blocks $t \times t$ on the diagonal.

Lemma 4.33. *Let $t \in T_I$ and let t_1, t_2 be its sons. Then*

$$S(t,t) = \begin{bmatrix} S(t_1,t_1) & S(t_1,t_2) \\ S(t_2,t_1) & S(t_2,t_2) + S(t_2,t_1)S(t_1,t_1)^{-1}S(t_1,t_2) \end{bmatrix}.$$

Proof. Let $S(t,t)$ be decomposed in the following way:

$$S(t,t) = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix}.$$

For the blocks (t_1,t_1), (t_1,t_2), and (t_2,t_1) the definition of r from (4.44) results in $r = \{i \in I : i < \min t\}$. Hence, we obtain

$$S(t_1,t_2) = A_{t_1t_2} - A_{t_1r}A_{rr}^{-1}A_{rt_2} = S_{12}.$$

Similarly, one sees that $S(t_1,t_1) = S_{11}$ and $S(t_2,t_1) = S_{21}$. It remains to show that

$$S_{22} = S(t_2,t_2) + S_{21}S_{11}^{-1}S_{12}.$$

For (t_2,t_2) the definition of r reads $r = \{i \in I : i < \min t_2\}$. Let $\bar{r} = \{i \in I : i < \min t\}$. Then from the definition of $S(t,t)$ it follows that

$$S(t_2,t_2) = A_{t_2t_2} - \begin{bmatrix} A_{t_2\bar{r}} & A_{t_2t_1} \end{bmatrix} \begin{bmatrix} A_{\bar{r}\bar{r}} & A_{\bar{r}t_1} \\ A_{t_1\bar{r}} & A_{t_1t_1} \end{bmatrix}^{-1} \begin{bmatrix} A_{\bar{r}t_2} \\ A_{t_1t_2} \end{bmatrix}.$$

Since

$$\begin{bmatrix} A_{\bar{r}\bar{r}} & A_{\bar{r}t_1} \\ A_{t_1\bar{r}} & A_{t_1t_1} \end{bmatrix}^{-1} = \begin{bmatrix} A_{\bar{r}\bar{r}}^{-1} & -A_{\bar{r}\bar{r}}^{-1}A_{\bar{r}t_1}S_{11}^{-1} \\ 0 & S_{11}^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -A_{t_1\bar{r}}A_{\bar{r}\bar{r}}^{-1} & I \end{bmatrix},$$

we have

$$\begin{aligned} S(t_2,t_2) &= A_{t_2t_2} - \begin{bmatrix} A_{t_2\bar{r}} & A_{t_2t_1} \end{bmatrix} \begin{bmatrix} A_{\bar{r}\bar{r}}^{-1} & -A_{\bar{r}\bar{r}}^{-1}A_{\bar{r}t_1}S_{11}^{-1} \\ 0 & S_{11}^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -A_{t_1\bar{r}}A_{\bar{r}\bar{r}}^{-1} & I \end{bmatrix} \begin{bmatrix} A_{\bar{r}t_2} \\ A_{t_1t_2} \end{bmatrix} \\ &= A_{t_2t_2} - \begin{bmatrix} A_{t_2\bar{r}} & A_{t_2t_1} \end{bmatrix} \begin{bmatrix} A_{\bar{r}\bar{r}}^{-1} & -A_{\bar{r}\bar{r}}^{-1}A_{\bar{r}t_1}S_{11}^{-1} \\ 0 & S_{11}^{-1} \end{bmatrix} \begin{bmatrix} A_{\bar{r}t_2} \\ S_{12} \end{bmatrix} \\ &= A_{t_2t_2} - \begin{bmatrix} A_{t_2\bar{r}}A_{\bar{r}\bar{r}}^{-1} & S_{21}S_{11}^{-1} \end{bmatrix} \begin{bmatrix} A_{\bar{r}t_2} \\ S_{12} \end{bmatrix} = S_{22} - S_{21}S_{11}^{-1}S_{12}, \end{aligned}$$

which proves the assertion. □

Since each block $t \times s$ in the upper triangular part, i.e., $\max t \le \min s$, can be embedded into the block $r \times r$, $r := \{i \in I : \min t \le i \le \max s\}$, Lemma 4.33 gives

$$S(t,s) = \begin{bmatrix} S(t_1,s) \\ S(t_2,s) + S(t_2,t_1)S(t_1,t_1)^{-1}S(t_1,s) \end{bmatrix}. \tag{4.45}$$

Similarly, for each block $t \times s$ in the lower triangular part, i.e., $\max s \le \min t$, it holds that

$$S(t,s) = \begin{bmatrix} S(t_1,s_1) & S(t_1,s_2) + S(t_1,s_1)S(s_1,s_1)^{-1}S(s_1,s_2) \\ S(t_2,s_1) & S(t_2,s_2) + S(t_2,s_1)S(s_1,s_1)^{-1}S(s_1,s_2) \end{bmatrix}.$$

4.3.2 Constructing the Factors $L_{\mathscr{H}}$ and $U_{\mathscr{H}}$

We assume that the corresponding Schur complement $S(t,s)$ for each admissible block $t \times s \in \mathscr{L}(T_{I\times I})$ can be approximated by a matrix of low rank with arbitrary accuracy; i.e, for all $\varepsilon > 0$ there is $\tilde{S}(t,s) \in \mathbb{R}^{t\times s}$ of rank $k_S \sim (\log n)^{\alpha} |\log \varepsilon|^{\beta}$ with some $\alpha, \beta > 0$ such that

$$\|S(t,s) - \tilde{S}(t,s)\|_2 \leq \varepsilon \|A\|_2. \tag{4.46}$$

According to Theorem 4.31 this assumption is fulfilled, for instance, in the case of finite element stiffness matrices.

In order to define the factors $L(t)$ and $U(t)$ of $S(t,t) = L(t)U(t)$, $t \in T_I \setminus \mathscr{L}(T_I)$, we set

$$L(t) := \begin{bmatrix} L(t_1) & 0 \\ S(t_2,t_1)U(t_1)^{-1} & L(t_2) \end{bmatrix} \quad \text{and} \quad U(t) := \begin{bmatrix} U(t_1) & L(t_1)^{-1}S(t_1,t_2) \\ 0 & U(t_2) \end{bmatrix}, \tag{4.47}$$

where

$$L(t_1)U(t_1) = S(t_1,t_1), \qquad L(t_2)U(t_2) = S(t_2,t_2),$$

and t_1, t_2 are the sons of t. If $t \in \mathscr{L}(T_I)$, then $L(t)$ and $U(t)$ are defined by the pointwise LU decomposition. Note that since

$$L(t)U(t) = \begin{bmatrix} L(t_1)U(t_1) & S(t_1,t_2) \\ S(t_2,t_1) & L(t_2)U(t_2) + S(t_2,t_1)S(t_1,t_1)^{-1}S(t_1,t_2) \end{bmatrix},$$

we obtain $L(t)U(t) = S(t,t)$ due to Lemma 4.33. The following lemma shows that the off-diagonal blocks in (4.47) can be approximated by hierarchical matrices. For its proof we will make use of

$$\|S(t,t)^{-1}\|_2 \leq \|A_{\hat{t}\hat{t}}^{-1}\|_2 \leq c\|A^{-1}\|_2, \tag{4.48}$$

where $\hat{t} := \{i \in I : i \leq \max t\}$. Estimate (4.48) follows from (4.39) and the fact that $S(t,t)^{-1}$ is the $t \times t$ sub-block of $A_{\hat{t}\hat{t}}^{-1}$.

Lemma 4.34. *Let X, Y solve $L(t)X = S(t,s)$ and $YU(t) = S(s,t)$, where $\max t \leq \min s$. Then X and Y can be approximated by $\tilde{X} \in \mathscr{H}(T_{t\times s}, k_S)$ and $\tilde{Y} \in \mathscr{H}(T_{s\times t}, k_S)$ such that*

$$\|X - \tilde{X}\|_2 \leq c\,\varepsilon L(T_I)\,\mathrm{cond}_2(A)\,\|U(t)\|_2$$

and

$$\|Y - \tilde{Y}\|_2 \leq c\,\varepsilon L(T_I)\,\mathrm{cond}_2(A)\,\|L(t)\|_2,$$

where $L(T_I)$ denotes the depth of the cluster tree T_I and the bound k_S on the block-wise rank was defined in (4.46).

Proof. By induction we first prove that X can be approximated by $\tilde{X} \in \mathscr{H}(T_{t\times s}, k_S)$ such that on each admissible sub-block $t' \times s'$ of $t \times s$ it holds that

$$\|X_{t's'} - \tilde{X}_{t's'}\|_2 \le c\,\varepsilon\,\mathrm{cond}_2(A)\,\|U(t')\|_2. \tag{4.49}$$

If $t \times s$ is an admissible leaf in T_I, then we have assumed (see (4.46)) that $S(t,s)$ can be approximated by a matrix $\tilde{S}(t,s) \in \mathbb{R}^{t\times s}$ of rank at most k_S. Hence, the rank of $\tilde{X} := L(t)^{-1}\tilde{S}(t,s)$ cannot exceed k_S and we have that

$$\begin{aligned}\|X - \tilde{X}\|_2 &= \|L(t)^{-1}[S(t,s) - \tilde{S}(t,s)]\|_2 = \|U(t)S(t,t)^{-1}[S(t,s) - \tilde{S}(t,s)]\|_2 \\ &\le \varepsilon\|S(t,t)^{-1}\|_2\|A\|_2\|U(t)\|_2 \le c\,\varepsilon\,\mathrm{cond}_2(A)\,\|U(t)\|_2\end{aligned}$$

due to (4.48). If $t \times s$ is not a leaf, then t has sons t_1, t_2 and s has sons s_1, s_2. Define $X_{11} \in \mathbb{R}^{t_1\times s_1}$, $X_{12} \in \mathbb{R}^{t_1\times s_2}$, $X_{21} \in \mathbb{R}^{t_2\times s_1}$, and $X_{22} \in \mathbb{R}^{t_2\times s_2}$ by

$$L(t_i)X_{ij} = S(t_i, s_j), \quad i,j = 1,2,$$

respectively. By induction we know that X_{11}, X_{12}, X_{21}, and X_{22} can be approximated by $\mathscr{H}$-matrices $\tilde{X}_{11}$, $\tilde{X}_{12}$, $\tilde{X}_{21}$, and $\tilde{X}_{22}$ to the sub-trees of $T_{I\times I}$ with roots $t_1 \times s_1$, $t_1 \times s_2$, $t_2 \times s_1$, and $t_2 \times s_2$, respectively. Hence,

$$X = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}$$

satisfies

$$\begin{aligned}L(t)X &= \begin{bmatrix} L(t_1) & 0 \\ S(t_2,t_1)U(t_1)^{-1} & L(t_2) \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \\ &= \begin{bmatrix} S(t_1,s) \\ S(t_2,s) + S(t_2,t_1)S(t_1,t_1)^{-1}S(t_1,s) \end{bmatrix} = S(t,s)\end{aligned}$$

due to the definition (4.47) of $L(t)$ and (4.45), and X can be approximated by

$$\tilde{X} := \begin{bmatrix} \tilde{X}_{11} & \tilde{X}_{12} \\ \tilde{X}_{21} & \tilde{X}_{22} \end{bmatrix} \in \mathscr{H}(T_{t\times s}, k_S)$$

satisfying (4.49). The assertion follows from Lemma 2.16, because

$$\begin{aligned}\|X - \tilde{X}\|_2 &\le c_{\mathrm{sp}} L(T_I) \max_{t'\times s' \in T_{t\times s}} \|X_{t's'} - \tilde{X}_{t's'}\|_2 \\ &\le c c_{\mathrm{sp}} \varepsilon L(T_I)\,\mathrm{cond}_2(A) \max_{t' \in T_t} \|U(t')\|_2 \\ &\le c c_{\mathrm{sp}} \varepsilon L(T_I)\,\mathrm{cond}_2(A)\,\|U(t)\|_2.\end{aligned}$$

The proof for Y can be done analogously. □

The following theorem is the main result of this section. Although the proved error estimates have a Wilkinson style, these theorems are not meant as estimates on the backward stability of an algorithm for the approximation by $\mathscr{H}$-matrices. However, they show that $\mathscr{H}$-matrix approximants exist and that their complexity depends logarithmically on the accuracy $\varepsilon \to 0$. A similar estimate obviously holds for the Cholesky decomposition.

Theorem 4.35. *Assume that* (4.46) *holds and let L and U be the uniquely determined unit lower and upper triangular factors of A. Then there are unit lower and upper triangular matrices $L_{\mathscr{H}}, U_{\mathscr{H}} \in \mathscr{H}(T_{I\times I}, k_S)$ such that*

$$\|A - L_{\mathscr{H}} U_{\mathscr{H}}\|_2 \leq c\,\varepsilon L(T_I)\,\mathrm{cond}_2(A)\,\|L\|_2\|U\|_2 + \mathscr{O}(\varepsilon^2),$$

where $L(T_I)$ denotes the depth of the tree T_I.

Proof. Since $A = S(I,I) = L(I)U(I)$, it follows from the uniqueness of the LU decomposition that $L = L(I)$ and $U = U(I)$. According to Lemma 4.34, there are $\mathscr{H}$-matrices $L_{\mathscr{H}}, U_{\mathscr{H}} \in \mathscr{H}(T_{I\times I}, k_S)$ satisfying

$$\|L - L_{\mathscr{H}}\|_2 \leq c\,\varepsilon L(T_I)\,\mathrm{cond}_2(A)\,\|L\|_2 \;\text{ and }\; \|U - U_{\mathscr{H}}\|_2 \leq c\,\varepsilon L(T_I)\,\mathrm{cond}_2(A)\,\|U\|_2.$$

As a consequence we have

$$\begin{aligned}\|A - L_{\mathscr{H}} U_{\mathscr{H}}\|_2 &\leq \|(L - L_{\mathscr{H}})U\|_2 + \|L(U - U_{\mathscr{H}})\|_2 + \|(L - L_{\mathscr{H}})(U - U_{\mathscr{H}})\|_2 \\ &\leq \|L - L_{\mathscr{H}}\|_2\|U\|_2 + \|L\|_2\|U - U_{\mathscr{H}}\|_2 + \|L - L_{\mathscr{H}}\|_2\|U - U_{\mathscr{H}}\|_2 \\ &\leq \left[2c\,\varepsilon L(T_I)\,\mathrm{cond}_2(A) + c^2\varepsilon^2 L^2(T_I)\,(\mathrm{cond}_2(A))^2\right]\|L\|_2\|U\|_2,\end{aligned}$$

which proves the assertion. □

We have seen in Theorem 4.31 that in the case of FE discretizations of elliptic partial differential operators of type (4.2) each Schur complement S in A can be approximated by an $\mathscr{H}$-matrix. Hence, the previous theorem can be applied to such matrices. Since the complexity depends logarithmically on the accuracy ε, the blockwise rank of the factors $L_{\mathscr{H}}$ and $U_{\mathscr{H}}$ compared with the rank of the inverse shows an additional logarithmic factor. However, from the following numerical experiments it will be seen that the complexity of the $\mathscr{H}$-matrix LU factorization is significantly smaller than the complexity of the $\mathscr{H}$-matrix inversion in practice. Further improvements can be obtained by applying the ideas of nested dissection to the LU factorization; see Sect. 4.5.

4.4 Numerical Experiments with the $\mathscr{H}$-Matrix LU Factorization

In this section we make use of the algorithms from Sect. 2.9 for computing approximate LU decompositions of FE stiffness matrices in two and three spatial

dimensions. The emphasis in these tests is laid on robustness with respect to varying coefficients of the underlying operator.

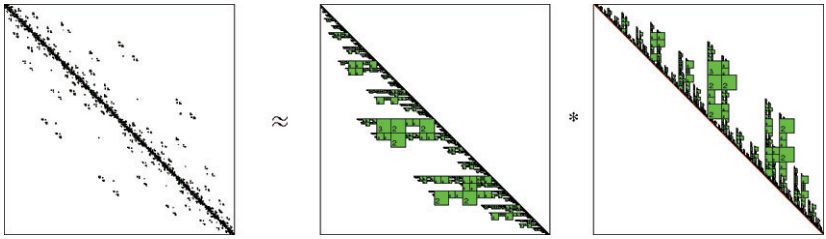


Fig. 4.7 Approximate *LU* decomposition.

In the first part of the experiments we apply the hierarchical *LU* decomposition to the same problems the hierarchical inverse was applied to. This allows a direct comparison of both structures. In the second part three-dimensional boundary value problems will be considered. We compare this approximate *LU* decomposition with recent multifrontal solvers. Additionally, low-precision approximations are used for preconditioning. Since the proposed preconditioner is explicit, it is particularly efficient if the same system with many right-hand sides has to be solved. All computations were carried out on an Athlon64 PC (2 GHz) with 12 GB of core memory. The minimal block size $n_{\min}$ was chosen 50 and for the cluster parameter $\eta = 1.1$ was used.

4.4.1 Two-Dimensional Diffusion

As a first set of tests we consider the Dirichlet boundary value problem from Sect. 4.1.6

$$\begin{aligned} -\operatorname{div} C(x)\nabla u &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned}$$

where $\Omega := (0,1) \times (0,1)$ is the unit square in $\mathbb{R}^2$ and C is the coefficient from (4.37). Note that the methods can be equally applied to other computational domains.

The main aim of these two-dimensional tests is to show that the computational complexity of the hierarchical *LU* decomposition is almost linear, thereby confirming our estimates from Sect. 4.3. In the following tables we compare the computational effort for decomposing the stiffness matrix of the above problem for different problem sizes n and for different amplitudes a. Since the discrete operator is symmetric positive definite, we actually compute the Cholesky decomposition LL^T.

Table 4.9 shows the required time in seconds, its memory consumption (MB), the maximum rank $k_{\max}$ of the blocks, and the backward error

$$\mathscr{E} := \frac{\|A - L_{\mathscr{H}} L_{\mathscr{H}}^T\|_2}{\|A\|_2}$$

for the amplitude $a = 1$. Apparently (see Fig. 4.8), the CPU time scales like $n \log^3 n$,

Table 4.9 $\mathscr{H}$-matrix Cholesky factorization for $a = 1$.

	$\varepsilon_{\mathscr{H}} = 1_{10}{-}2$				$\varepsilon_{\mathscr{H}} = 1_{10}{-}6$			
n	time	MB	$k_{\max}$	$\mathscr{E}$	time	MB	$k_{\max}$	$\mathscr{E}$
38 025	1.4s	31	14	$2.3_{10}{-}3$	2.4s	43	17	$2.8_{10}{-}7$
65 025	2.9s	51	15	$1.4_{10}{-}3$	5.4s	77	15	$2.4_{10}{-}7$
129 600	6.7s	109	13	$2.5_{10}{-}3$	12.7s	168	16	$3.2_{10}{-}7$
278 784	16.6s	252	10	$1.8_{10}{-}3$	33.5s	400	14	$2.9_{10}{-}7$
529 984	37.3s	502	13	$2.5_{10}{-}3$	77.2s	820	16	$3.7_{10}{-}7$
1 183 744	101.2s	1212	11	$2.0_{10}{-}3$	229.5s	2021	14	$3.3_{10}{-}7$
2 310 400	253.9s	2474	13	$2.7_{10}{-}3$	688.2s	4236	16	$2.3_{10}{-}7$
4 473 225	585.1s	4902	11	$2.5_{10}{-}3$	1881.8s	8659	14	$3.9_{10}{-}7$

while the storage requirement has complexity $n \log n$. The backward error seems to be independent of n. Increasing the rounding precision $\varepsilon_{\mathscr{H}}$ directly results in a smaller backward error. Table 4.10 shows the same quantities for $a = 10^9$ with only slight changes of the results. In fact, the results become slightly better. The depen-

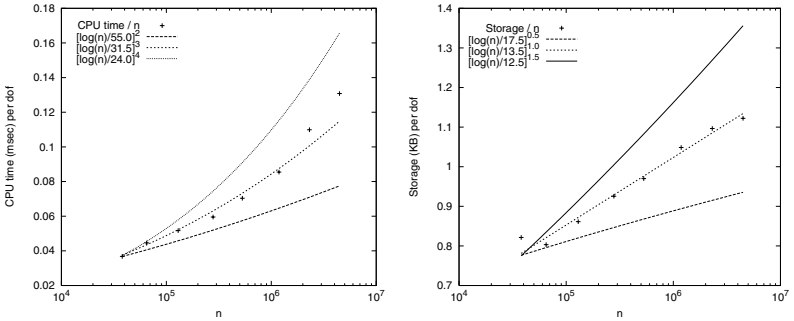


Fig. 4.8 CPU time and storage compared with $n \log^3 n$ and $n \log n$.

dence of both the computational effort and the backward error on the amplitude a is surprisingly weak demonstrating the robustness of this approximate LU decomposition. Compared with the hierarchical inverse, the complexity of the hierarchical LU decomposition is more than 10 to 15 times faster and needs three times less storage.

Table 4.10 $\mathcal{H}$-matrix Cholesky decomposition for $a = 10^9$.

	$\varepsilon_{\mathcal{H}} = 1_{10}-2$				$\varepsilon_{\mathcal{H}} = 1_{10}-6$			
n	time	MB	$k_{\max}$	$\mathcal{E}$	time	MB	$k_{\max}$	$\mathcal{E}$
38 025	1.2s	29	14	$1.5_{10}-03$	2.1s	39	17	$7.4_{10}-09$
65 025	2.5s	47	15	$2.1_{10}-04$	4.3s	66	18	$4.4_{10}-10$
129 600	5.9s	98	13	$1.3_{10}-03$	11.1s	150	18	$4.9_{10}-10$
278 784	14.3s	226	10	$2.2_{10}-03$	26.8s	344	18	$7.5_{10}-10$
529 984	32.1s	438	14	$1.3_{10}-03$	65.5s	720	20	$3.4_{10}-08$
1 183 744	82.7s	1072	11	$2.5_{10}-03$	170.6s	1718	28	$8.3_{10}-08$
2 310 400	223.9s	2119	14	$2.5_{10}-03$	515.9s	3659	29	$1.5_{10}-09$
4 473 225	555.3s	4275	12	$2.6_{10}-03$	1068.7s	7215	28	$9.6_{10}-08$

Table 4.11 shows the CPU time required to solve linear systems by forward/backward substitution. The results correspond to the amplitude $a = 10^9$. Again, the CPU time scales almost linearly as estimated in Sect. 2.9.

Table 4.11 Forward/backward substitution.

	$\varepsilon_{\mathcal{H}} = 1_{10}-2$	$\varepsilon_{\mathcal{H}} = 1_{10}-6$
n	time	time
38 025	0.04s	0.05s
65 025	0.06s	0.08s
129 600	0.13s	0.18s
278 784	0.29s	0.41s
529 984	0.57s	0.85s
1 183 744	1.37s	2.01s
2 310 400	3.43s	5.03s
4 473 225	6.75s	10.42s

4.4.1.1 Preconditioning

We know from Sect. 2.12 that problem-independent convergence rates can be obtained if low-accuracy approximations are used for preconditioning. In the following tables we compare the diagonally preconditioned conjugate gradient method (DPCG) and the $\mathcal{H}$-matrix *LU* decomposition preconditioned CG ($\mathcal{H}$PCG) method. The relative accuracy of the residual is $1_{10}-6$. Since the accuracy of the $\mathcal{H}$-matrix *LU* decomposition can be quite low for preconditioning, it is sufficient to compute the preconditioner in single precision arithmetic, which saves half of the memory.

In Table 4.12 the amplitude a of the coefficient $C(x)$ is chosen 100. The time needed to compute the approximate Cholesky decomposition is shown in the third column for increasing problem sizes n. The memory consumption can be found in the fourth column. The number of iterations and the CPU time of $\mathcal{H}$PCG are compared with the respective quantities for DPCG in the last four columns. $\varepsilon_{\mathcal{H}}$

denotes the rounding precision which was used when decomposing the stiffness matrix. According results are shown in Tables 4.13 and 4.14 for the amplitudes

Table 4.12 $\mathscr{H}$PCG and DPCG for $a = 10^2$.

		$\mathscr{H}$PCG				DPCG	
n	$\varepsilon_{\mathscr{H}}$	time	MB	#It	time	#It	time
38 025	$8.7_{10}-1$	0.6s	12	34	0.9s	1153	10s
65 025	$5.0_{10}-1$	1.1s	18	35	1.6s	1408	22s
129 600	$6.0_{10}-2$	3.3s	43	36	3.4s	2033	63s
278 784	$4.0_{10}-2$	8.4s	109	29	6.3s	2995	210s
529 984	$3.7_{10}-2$	19.9s	215	41	17.6s	4127	585s
1 183 744	$2.0_{10}-2$	52.8s	545	34	43.7s	6279	2327s
2 310 400	$1.0_{10}-2$	133.1s	1212	37	80.2s	8567	5988s
4 473 225	$6.0_{10}-3$	430.7s	2578	39	214.7s	–	–

$a = 10^4$ and $a = 10^6$, respectively. The iterations of DPCG were stopped after 120 minutes.

Table 4.13 $\mathscr{H}$PCG and DPCG for $a = 10^4$.

		$\mathscr{H}$PCG				DPCG	
n	$\varepsilon_{\mathscr{H}}$	time	MB	#It	time	#It	time
38 025	$8.7_{10}-1$	0.6s	12	35	0.9s	4286	38s
65 025	$5.0_{10}-1$	1.1s	18	38	1.6s	5192	82s
129 600	$6.0_{10}-2$	3.3s	43	37	3.5s	7327	229s
278 784	$4.0_{10}-2$	8.4s	109	30	6.5s	11241	783s
529 984	$3.7_{10}-2$	20.4s	215	41	17.5s	15353	2279s
1 183 744	$2.0_{10}-2$	50.9s	545	37	37.9s	–	–
2 310 400	$1.0_{10}-2$	133.8s	1204	41	95.8s	–	–
4 473 225	$6.0_{10}-3$	430.4s	2564	53	290.2s	–	–

Table 4.14 $\mathscr{H}$PCG and DPCG for $a = 10^6$.

		$\mathscr{H}$PCG				DPCG	
n	$\varepsilon_{\mathscr{H}}$	time	MB	#It	time	#It	time
38 025	$8.7_{10}-1$	0.6s	12	38	1.0s	–	–
65 025	$5.0_{10}-1$	1.1s	18	41	1.8s	–	–
129 600	$6.0_{10}-2$	3.3s	43	39	3.7s	–	–
278 784	$4.0_{10}-2$	8.4s	109	30	6.5s	–	–
529 984	$3.7_{10}-2$	19.8s	215	41	17.5s	–	–
1 183 744	$2.0_{10}-2$	55.1s	545	39	47.3s	–	–
2 310 400	$1.0_{10}-2$	135.1s	1212	45	108.5s	–	–
4 473 225	$6.0_{10}-3$	406.7s	2574	56	309.4s	–	–

Apparently, the diagonal preconditioner suffers from both the amplitude a and the number of degrees of freedom n. The number of iterations of $\mathscr{H}$PCG, however, is almost constant. For this purpose, the rounding precision $\varepsilon_{\mathscr{H}}$ has to be increased with increasing n since the condition number appears in Lemma 2.43, which grows with n. The $\mathscr{H}$-matrix preconditioner is able to adapt itself to the varying coefficients without significant changes of its complexity. The usage of the $\mathscr{H}$-matrix *LU* decomposition as a preconditioner is more efficient than using it as a direct solver.

4.4.1.2 Diffusion through a Skin-Like Domain

Although Neumann boundary conditions have not been treated in this chapter, we consider the following problem

$$\begin{aligned} -\operatorname{div}\alpha(x)\nabla u &= 0 && \text{in } \Omega,\\ u &= 0 && \text{on } \Gamma_D^-,\\ u &= +1 && \text{on } \Gamma_D^+,\\ \partial_\nu u &= 0 && \text{on } \Gamma_N, \end{aligned}$$

where $\Omega = (0,1)^2$ and Γ_N is the union of the left and the right boundary of the geometry from Fig. 4.9. Γ_D^+ is the upper and Γ_D^- the lower boundary. The diffusion coefficient α is set to a in the dark parts of Fig. 4.9 and to 1 in the remaining domain. This problem is taken from [159] and corresponds to the penetration of a drug through the skin. Hence, α has the meaning of a permeability, which is high in the lipid structure and low in the cells. While in [159] a preconditioning method based on substructuring is used for solving this problem, we will simply apply the approximate *LU* decomposition for preconditioning without taking into account the coefficient α. We will use the parameter b to check whether the complexity depends on the thickness of the lipid structure. Table 4.15 shows the time needed to construct the $\mathscr{H}$-matrix Cholesky preconditioner with rounding precision $\varepsilon_{\mathscr{H}} = 0.007$, its memory consumption, the number of iterations of PCG, and the time required for a residual error below $1_{10}-5$. The geometry has 60 rows each containing 3 cells. We conclude from these tests that the hierarchical *LU* preconditioner is robust with

Table 4.15 PCG for $n = 499\,848$.

	$a = 1_{10}-3$				$a = 1_{10}-6$			
b	time	MB	#It	time	time	MB	#It	time
0.1	13.0s	160	39	15.3s	12.4s	154	35	13.6s
0.05	11.4s	151	31	12.2s	10.7s	140	29	10.8s
0.01	16.1s	179	33	13.9s	15.6s	175	28	11.5s

respect to nonsmooth coefficients and large aspect ratios.

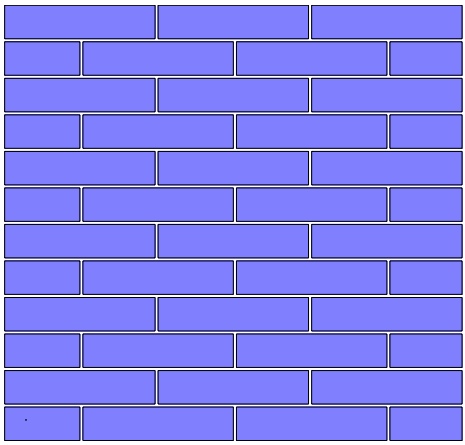


Fig. 4.9 Geometry of skin fragment with diffusion coefficient.

4.4.2 Convection-Diffusion Problems

In the next tests, operators of type

$$\mathscr{L} = -\Delta + \beta \cdot \nabla$$

are considered. The convection coefficient β is randomly chosen, i.e., $\beta(x) \in [-a,a]^2$ for each $x \in \Omega = (0,1) \times (0,1)$. Since the standard FE method becomes unstable for large a, i.e., the stiffness matrix may become singular, we have restricted ourselves to the cases $a = 10, 100$. In the first tests we use the hierarchical *LU* decomposition as a direct solver. Table 4.16 contains the results for the case $a = 10$. Note that a symmetry of the stiffness matrix could not be exploited for this kind of problems. Table 4.17 contains the respective results for $a = 100$.

Although the computational effort has increased compared with the diffusion problem, it still scales almost linearly. Again, a dependence on the coefficient β can hardly be observed.

Table 4.18 shows the time and the amount of memory needed to compute the exact *LU* decomposition using SuperLU [78]. Both methods seem to scale in a similar way with n, while SuperLU is 6.4 times faster for the smallest n and 2.5 times faster for the largest n in our tests. The direct solution using the hierarchical *LU* decomposition does not pay for two-dimensional problems. Here, multifrontal solvers are

Table 4.16 $\mathscr{H}$-matrix *LU* decomposition for $a = 10$.

	$\varepsilon_{\mathscr{H}} = 1_{10}-2$				$\varepsilon_{\mathscr{H}} = 1_{10}-6$			
n	time	MB	$k_{\max}$	$\mathscr{E}$	time	MB	$k_{\max}$	$\mathscr{E}$
39 061	4.5s	65	14	$1.1_{10}-3$	7.1s	92	17	$2.6_{10}-7$
78 961	10.5s	136	10	$1.5_{10}-3$	17.8s	201	13	$1.4_{10}-7$
159 201	25.7s	269	14	$1.7_{10}-3$	43.8s	447	17	$3.9_{10}-7$
318 096	61.2s	618	10	$1.5_{10}-3$	112.8s	969	13	$2.3_{10}-7$
638 401	152.3s	1319	14	$1.7_{10}-3$	318.5s	2110	17	$4.1_{10}-7$
1 276 900	356.9s	2728	10	$2.5_{10}-2$	1 103.9s	4515	13	$2.3_{10}-7$

Table 4.17 $\mathscr{H}$-matrix *LU* decomposition for $a = 100$.

	$\varepsilon_{\mathscr{H}} = 1_{10}-2$				$\varepsilon_{\mathscr{H}} = 1_{10}-6$			
n	time	MB	$k_{\max}$	$\mathscr{E}$	time	MB	$k_{\max}$	$\mathscr{E}$
39 061	4.5s	65	14	$1.3_{10}-3$	7.1s	92	17	$3.5_{10}-7$
78 961	10.4s	136	10	$1.6_{10}-3$	18.0s	201	13	$2.6_{10}-7$
159 201	25.6s	296	14	$1.7_{10}-3$	43.0s	447	17	$3.6_{10}-7$
318 096	59.3s	618	10	$1.5_{10}-3$	112.3s	969	13	$2.6_{10}-7$
638 401	152.3s	1318	14	$1.8_{10}-3$	315.4s	2109	17	$3.9_{10}-7$
1 276 900	357.4s	2726	10	$1.6_{10}-3$	1 120.6s	4514	13	$2.4_{10}-7$

Table 4.18 *LU* decomposition using SuperLU.

n	time	MB
39 061	0.7s	43
78 961	2.2s	96
159 201	6.4s	217
318 096	17.6s	461
638 401	57.8s	1 051
1 276 900	142.5s	2 145

well developed and provide an exact and robust solution. However, the $\mathscr{H}$-matrix *LU* decomposition is attractive for preconditioning.

4.4.2.1 Preconditioning

The number of iterations for different parameters a are shown in Table 4.19. The third and seventh column contain the time used to construct the preconditioner while columns six and ten show the CPU time required by preconditioned GMRes.

4.4.3 Three-Dimensional Diffusion

As we have seen in Chap. 1, the structure of $\mathscr{H}$-matrices can be equally applied to any quasi-uniform FE discretization of Ω given just by the grid information.

Table 4.19 $\mathscr{H}$-matrix *LU* preconditioned GMRes for convection-diffusion problems.

		$a=10$				$a=100$			
n	$\varepsilon_{\mathscr{H}}$	time	MB	#It	time	time	MB	#It	time
39 601	$7.2_{10}-1$	3.3s	27	16	0.6s	3.1s	26	30	1.1s
78 961	$7.4_{10}-1$	6.8s	55	22	1.7s	6.8s	55	30	2.5s
159 201	$2.3_{10}-1$	18.7s	120	28	4.6s	18.8s	120	28	4.6s
318 096	$6.0_{10}-2$	39.1s	258	24	9.0s	39.1s	258	30	10.4s
638 401	$3.9_{10}-2$	108.3s	582	30	26.5s	108.5s	584	29	24.0s
1 276 900	$3.9_{10}-2$	207.0s	1206	14	22.3s	214.4s	1191	28	49.2s

In order to demonstrate that the $\mathscr{H}$-matrix *LU* decomposition is also efficient for three-dimensional problems, we test the proposed method on two tetrahedral discretizations ($n = 147\,339$ and $n = 1\,237\,831$ of unknowns) of the volume shown in Fig. 4.10. The meshes were generated using NETGEN [233].

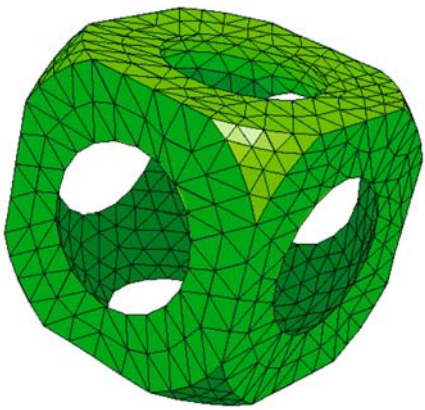


Fig. 4.10 The computational domain.

In the next experiments we compare the numerical effort for generating a hierarchical Cholesky decomposition of the stiffness matrices arising from the Dirichlet boundary value problem

$$\begin{aligned} -\operatorname{div} C(x)\nabla u &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned}$$

where $C(x) \in \mathbb{R}^{3\times 3}$ is a symmetric positive definite matrix for all $x \in \Omega$. The coefficients c_{ij}, $i,j = 1,2,3$, are set to one in the left half-space and to a random number from the interval $[0, 10^3]$ in each point of the right half-space.

In the second column of Table 4.20 the time that was needed to compute the approximate Cholesky decomposition is shown for different rounding precisions $\varepsilon_{\mathscr{H}}$.

The memory consumption can be found in the third column. Columns four and five contain the maximum rank among the blocks and the backward error. In the last row according results obtained by SuperLU are shown. Compared with the two-

Table 4.20 $\mathscr{H}$-matrix Cholesky decomposition for three spatial dimensions.

	$n = 147\,339$				$n = 1\,237\,831$			
$\varepsilon_{\mathscr{H}}$	time	MB	$k_{\max}$	$\mathscr{E}$	time	MB	$k_{\max}$	$\mathscr{E}$
$1_{10}-1$	20.6s	97	86	$3.5_{10}-2$	458.9s	1 222	115	$4.0_{10}-2$
$1_{10}-2$	30.6s	133	98	$4.7_{10}-3$	782.4s	1 858	127	$5.5_{10}-3$
$1_{10}-4$	52.2s	202	106	$6.0_{10}-5$	1 701.0s	3 251	134	$5.5_{10}-5$
$1_{10}-6$	78.8s	266	107	$4.2_{10}-7$	2 950.8s	4 689	134	$4.8_{10}-7$
SuperLU	142.5s	905		$1.5_{10}-14$	–	–		–

dimensional problems from Sect. 4.4.1, the CPU time for decomposing the matrix has increased. However, a similar behavior as in the two-dimensional tests can be observed: The computational complexity scales almost linearly and the backward error does not seem to depend on n. Again, the proposed method is able to adapt itself to the varying coefficients. The comparison with SuperLU shows that SuperLU is slower and requires more storage. The decomposition algorithm ran out of memory when it was applied to the larger problem. Here, the higher asymptotic complexity of SuperLU is revealed. The proposed approximate *LU* decomposition still scales linearly and therefore becomes more efficient the larger n is. This is especially true for low-precision approximations.

Table 4.21 contains the time required to solve a linear system by forward/backward substitution. The third and fifth column contain the error

$$\mathscr{E}_x := \frac{\|\tilde{x} - x\|_2}{\|x\|_2}$$

of the solution vector $\tilde{x}$ compared with the exact solution x. These tests show that

Table 4.21 Forward/backward substitution for in $\mathbb{R}^3$.

	$n = 147\,339$		$n = 1\,237\,831$	
$\varepsilon_{\mathscr{H}}$	time	$\mathscr{E}_x$	time	$\mathscr{E}_x$
$1_{10}-1$	0.1s	$6.2_{10}-02$	1.6s	$2.7_{10}-1$
$1_{10}-2$	0.2s	$8.7_{10}-03$	2.8s	$5.4_{10}-2$
$1_{10}-4$	0.3s	$2.9_{10}-05$	4.2s	$2.6_{10}-4$
$1_{10}-6$	0.4s	$2.3_{10}-07$	6.3s	$1.3_{10}-6$
SuperLU	0.8s	$1.4_{10}-14$	–	–

an approximate *LU* decomposition can be computed in the set of $\mathscr{H}$-matrices with logarithmic-linear complexity. The approximation is robust with respect to large and nonsmooth coefficients. The comparison with a recent direct solver shows that

a direct solution using the approximate *LU* decomposition pays for the considered problem sizes in three spatial dimensions.

4.4.3.1 Preconditioning

Instead of using the $\mathscr{H}$-matrix *LU* decomposition as a direct solver, it is advantageous to use it as a preconditioner. In order to be able to compare the numerical effort with the costs of the direct solver, we test the preconditioner on the two tetrahedral discretizations ($n = 147\,339$ and $n = 1\,237\,831$ of unknowns) of the volume shown in Fig. 4.10.

Table 4.22 shows the time needed to compute the preconditioner, its memory consumption, the number of iterations, and the time required to obtain a relative residual below $1_{10}-6$. The parameter a denotes the amplitude of the randomly chosen coefficients. For computing the preconditioner, the rounding precision $\varepsilon_{\mathscr{H}}$ was set to 0.8.

Table 4.22 Iterative solution using $\mathscr{H}$-matrix Cholesky preconditioner.

	$a = 10^2$				$a = 10^6$			
n	time	MB	#It	time	time	MB	#It	time
147 339	4.0s	42	27	3.0s	4.0s	42	33	3.6s
1 237 831	82.4s	494	51	58.4s	81.8s	494	62	72.4s

The same observation as in the previous tests can be made: The number of iterations depends neither on the amplitude a nor on the discretization parameter n. Furthermore, the computational effort scales almost linearly with respect to n.

4.5 Nested Dissection *LU* Factorization

It is well-known that the *LU* decomposition suffers from fill-in. This effect is reduced by nested dissection reorderings [99, 144], which recursively subdivide the index set I into clusters separated by separator clusters; see Remark 1.30. In this section we will adopt the ideas of nested dissection *LU* factorizations to hierarchical matrices, thereby improving the efficiency of hierarchical *LU* factorizations of finite element discretizations. A major advantage of nested dissection is that due to the idea of separation it allows to parallelize the *LU* factorization algorithm.

The ideas of nested dissection can be applied to both geometrical (see [213]) and algebraic clustering. Since the efficiency of the whole method depends significantly on the size of the separators and since the algebraic approach is closer to the actual properties of the finite element matrix A, we favor the latter kind of matrix partitioning for this purpose.

4.5.1 Matrix Partitioning

Our aim is to construct a nested dissection block cluster tree $T_{I\times I}$ of $I \times I$ based on the matrix graph (see Definition 1.14) of A. We adapt the usual definition (see Definition 1.16) of a cluster tree to the special situation of nested dissection. A cluster tree T_I is called a **nested dissection cluster tree** for an index set I if it satisfies the following conditions:

(i) I is the root of T_I;
(ii) if $t \in T_I$ is not a leaf and if it is not a separator cluster, then t is a disjoint union of its sons $S_I(t) = \{t_1, t_{\text{sep}}, t_2\} \subset T_I$. Furthermore, it holds that $A_{t_1 t_2} = 0 = A_{t_2 t_1}$;
(iii) if $t \in T_I$ is not a leaf and if it is a separator cluster, then t is a disjoint union of its sons $S_I(t) = \{t_1, t_2\} \subset T_I$.

Due to (ii), the matrix A_{tt} for a cluster $t \in T_I \setminus \mathscr{L}(T_I)$ that is not a separator has the structure

$$A_{tt} = \begin{bmatrix} A_{t_1 t_1} & & A_{t_1 t_{\text{sep}}} \\ & A_{t_2 t_2} & A_{t_2 t_{\text{sep}}} \\ A_{t_{\text{sep}} t_1} & A_{t_{\text{sep}} t_2} & A_{t_{\text{sep}} t_{\text{sep}}} \end{bmatrix},$$

i.e., the cluster t_{sep} separates t_1 from t_2.

The construction of nested dissection cluster trees can be done by first constructing a usual cluster tree based on spectral bisection as explained in Sect. 1.4.1.3. After that, for each cluster t and its two sons t_1' and t_2' one computes the vertex set t_{sep}. To this end consider the matrix graph $G_t = (t, E_t)$ of A_{tt}, where

$$E_t := \{(i,j) \in t \times t : a_{ij} \neq 0 \text{ or } a_{ji} = 0\}.$$

Furthermore, we define the boundaries

$$\begin{aligned} \partial t_1' &:= \{u \in t_1' : \exists v \in t_2' \text{ such that } (u,v) \in E_t\}, \\ \partial t_2' &:= \{v \in t_2' : \exists u \in t_1' \text{ such that } (v,u) \in E_t\} \end{aligned}$$

of t_1' and t_2' and the edge set $E_{12} := \{(u,v) \in E_t : u \in \partial t_1', v \in \partial t_2'\}$ between $\partial t_1'$ and $\partial t_2'$. To get a small separator t_{sep}, the **minimal vertex cover algorithm** [147] is applied to the bipartite graph $(\partial t_1' \cup \partial t_2', E_{12})$. Finally, the vertices belonging to the minimal vertex cover are moved out of t_1' and t_2' to t_{sep}. This generates the desired partition $\{t_1, t_{\text{sep}}, t_2\}$ of t. The construction is recursively applied to t_1 and t_2, while t_{sep} is recursively subdivided using spectral bisection.

Having constructed a nested dissection cluster tree from the matrix graph of the sparse matrix A, we can compute the block cluster tree $T_{I\times I}$. Since we want to use the algebraic way of matrix partitioning, according to the results of Sect. 4.1.3 for the construction of the latter tree one has to take into account the algebraic admissibility condition

$$\min\{\operatorname{diam}(t), \operatorname{diam}(s)\} \leq \eta \operatorname{dist}(t,s). \tag{4.50}$$

The difficulty with this condition is that a naive evaluation would spoil the overall complexity. In [25] a multi-level Dijkstra algorithm is presented which computes

approximations to the expressions appearing in (4.50) in such a way that the overall logarithmic-linear complexity is preserved. A block cluster tree can hence be constructed using the descendant mapping $S_{I\times I}$ defined by

$$S_{I\times I}(t,s) := \begin{cases} \emptyset, & \text{if } S_I(t) = \emptyset \text{ or } S_I(s) = \emptyset, \\ \emptyset, & \text{if } t \neq s \text{ and neither } t \text{ nor } s \text{ are separators}, \\ \emptyset, & \text{if } t \text{ or } s \text{ are separators and satisfy (4.50)}, \\ S_I(t) \times S_I(s), & \text{else.} \end{cases}$$

Notice that we have already proved the boundedness of c_{sp} and c_{id} for the algebraic admissibility condition (4.50); see Example 1.38 and Lemma 2.29. Since the complexity analysis of the hierarchical matrix arithmetic in Chap. 2 is based on the latter constants, the algebraic partition leads to the same order of complexity as the geometric one.

4.5.2 Approximation of the Factors of the *LU* Decomposition

The existence of $\mathscr{H}$-matrix approximants to the inverse of A is important for theoretical purposes. In this chapter we have already seen that for practical purposes

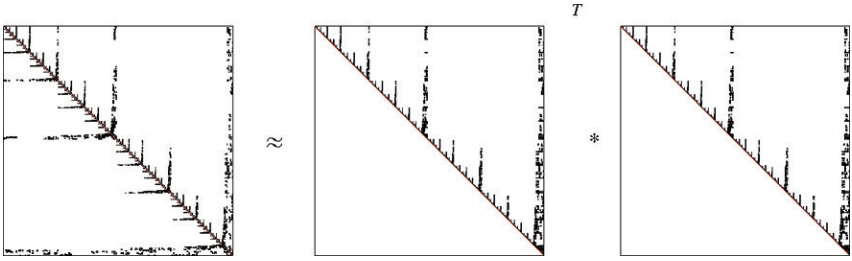


Fig. 4.11 Nested dissection Cholesky decomposition.

one is better off using $\mathscr{H}$-matrix approximations to the factors of the LU decomposition. We want to remind the reader of Theorem 4.35, which states that the local rank of an $\mathscr{H}$-matrix approximation of the factors of the LU decomposition can be related to the rank of the approximation of Schur complements in a matrix on the same partition. In particular this result remains valid for nested dissection partitions. Hence, as long as we can guarantee that Schur complements of sub-blocks $b \in \mathscr{L}(T_{I\times I})$ in A can be approximated with logarithmic rank, we know that the factors of the LU decomposition can be approximated with logarithmic-linear complexity. Therefore, in the rest of this section we concentrate on the approximation of the Schur complements.

Let A be partitioned as in (4.38). We will show that each block $b = t \times s$ of the Schur complement

$$S := A_{r'r'} - A_{rr'}^T A_{rr}^{-1} A_{rr'}$$

satisfying (4.50) can be approximated by a matrix of rank k, where k depends logarithmically on both the approximation accuracy and n. The proofs will use arguments similar to those used in Sect. 4.2 in the case of geometric matrix partitions.

Let $t \in T_I$ be a cluster. By

$$\mathcal{N}(t) = \{i \in I : \operatorname{dist}(i,t) \le 1\} \quad \text{and} \quad \mathcal{F}_\eta(t) = \{i \in I : \eta \operatorname{dist}(i,t) \ge \operatorname{diam}(t)\} \tag{4.51}$$

we denote the neighborhood and the **far-field** of t. The following lemma states that the neighborhood of t is in the far-field of the neighborhood of s if t is in the far-field of s.

Lemma 4.36. *Let* $\eta > 0$ *and* $\operatorname{diam} s \ge 4(1+\eta)$. *If* $t \subset \mathcal{F}_\eta(s)$, *then* $\mathcal{N}(t) \subset \mathcal{F}_{2\eta}(\mathcal{N}(s))$.

Proof. Since $\operatorname{diam} \mathcal{N}(s) \le \operatorname{diam} s + 2$, we obtain

$$\begin{aligned}
\eta \operatorname{dist}(\mathcal{N}(t), \mathcal{N}(s)) &\ge \eta \operatorname{dist}(t,s) - 2\eta > \operatorname{diam} s - 2\eta \\
&\ge \left(1 - \frac{2}{\operatorname{diam} \mathcal{N}(s)}(1+\eta)\right) \operatorname{diam} \mathcal{N}(s) \\
&\ge \frac{1}{2} \operatorname{diam} \mathcal{N}(s),
\end{aligned}$$

which proves the assertion. □

Using the previous lemma, we can prove that the Schur complement S of each block in A can be approximated. For this purpose we have to assume that there is a constant $c > 0$ such that

$$\|A_{rr}^{-1}\|_2 \le c \|A^{-1}\|_2 \tag{4.52}$$

for all $r \in T_I$. Additionally, motivated by Theorem 4.15 and Theorem 4.16, we assume that for each $r \subset I$ there is $C_{\mathcal{H}} \in \mathcal{H}(T_{r\times r}, k)$, $k \sim |\log \varepsilon|^m$, such that

$$\|A_{rr}^{-1} - C_{\mathcal{H}}\|_2 \le \varepsilon \|A_{rr}^{-1}\|_2. \tag{4.53}$$

Theorem 4.37. *Let* $A \in \mathbb{R}^{n\times n}$ *be partitioned as in* (4.38). *Then for the Schur complement*

$$S = A_{r'r'} - A_{rr'}^T A_{rr}^{-1} A_{rr'}$$

of A_{rr} *in* A *and all* $\varepsilon > 0$ *there is* $S_{\mathcal{H}} \in \mathcal{H}(T_{I\times I}, k)$, *where* $k \sim |\log \varepsilon|^m$, *such that*

$$\|S - S_{\mathcal{H}}\|_2 < (\log n)\, \varepsilon \operatorname{cond}_2(A) \|A\|_2.$$

Proof. We have to show that for each admissible block $t \times s \in \mathcal{L}(T_{I\times I})$ from the rows and columns of $A_{r'r'}$ and any prescribed accuracy $\varepsilon > 0$ we can find a low-rank

matrix which approximates S_{ts} with accuracy ε. Since $t \times s$ is admissible, $(A_{r'r'})_{ts} = 0$ holds. Hence,

$$S_{ts} = -A_{tr}A_{rr}^{-1}A_{rs} = -\sum_{i,j\in r} A_{ti}(A_{rr}^{-1})_{ij}A_{js}.$$

If $i \notin \mathcal{N}(t)$, where $\mathcal{N}(t)$ is defined in (4.51), then $A_{ti} = 0$. If on the other hand $j \notin \mathcal{N}(s)$, then $A_{js} = 0$. With the notation $\mathcal{N}'(t) := \mathcal{N}(t) \cap r$, we have

$$S_{ts} = -\sum_{i\in\mathcal{N}'(t),\, j\in\mathcal{N}'(s)} A_{ti}(A_{rr}^{-1})_{ij}A_{js}.$$

Since $t \times s$ is admissible, $t \subset \mathcal{F}_\eta(s)$ or $s \subset \mathcal{F}_\eta(t)$ holds. According to Lemma 4.36, it follows that $\mathcal{N}(t) \subset \mathcal{F}_{2\eta}(\mathcal{N}(s))$ or $\mathcal{N}(s) \subset \mathcal{F}_{2\eta}(\mathcal{N}(t))$ is valid. According to (4.53) (with η replaced by 2η), there is $U \in \mathbb{R}^{\mathcal{N}'(t)\times k}$ and $V \in \mathbb{R}^{\mathcal{N}'(s)\times k}$ with $k \sim |\log\varepsilon|^d$ such that

$$\|(A_{rr}^{-1})_{\mathcal{N}'(t)\mathcal{N}'(s)} - UV^T\|_2 < \varepsilon\|A_{rr}^{-1}\|_2.$$

Let U and V be extended to $\hat{U} \in \mathbb{R}^{r\times k}$ and $\hat{V} \in \mathbb{R}^{r\times k}$ by adding zero rows. Observe that

$$\begin{aligned} A_{tr}\hat{U}\hat{V}^T A_{rs} &= \sum_{i\in\mathcal{N}'(t),\, j\in\mathcal{N}'(s)} \sum_{\ell=1}^{k} A_{ti}U_{i\ell}V_{j\ell}A_{js} \\ &= \sum_{\ell=1}^{k} \left(\sum_{i\in\mathcal{N}'(t)} A_{ti}U_{i\ell}\right)\left(\sum_{j\in\mathcal{N}'(s)} V_{j\ell}A_{js}\right) = XY^T, \end{aligned}$$

where $X \in \mathbb{R}^{t\times k}$, $Y \in \mathbb{R}^{s\times k}$ with the entries

$$X_{t\ell} := \sum_{i\in\mathcal{N}'(t)} A_{ti}U_{i\ell} \quad \text{and} \quad Y_{s\ell} := \sum_{j\in\mathcal{N}'(s)} V_{j\ell}A_{js}, \qquad \ell = 1,\ldots,k.$$

Define $B \in \mathbb{R}^{r\times r}$ with entries

$$b_{ij} = \begin{cases} (A_{rr}^{-1})_{ij}, & \text{if } i \in \mathcal{N}'(t) \text{ and } j \in \mathcal{N}'(s), \\ 0, & \text{else,} \end{cases}$$

then using (4.52)

$$\begin{aligned} \|S_b - XY^T\|_2 = \|A_{tr}(B - \hat{U}\hat{V}^T)A_{rs}\|_2 &\le \|A_{tr}\|_2\|(A_{rr}^{-1})_{\mathcal{N}'(t)\mathcal{N}'(s)} - UV^T\|_2\|A_{rs}\|_2 \\ &< \|A_{tr}\|_2\|A_{rr}^{-1}\|_2\|A_{rs}\|_2 \le \varepsilon\,\mathrm{cond}_2(A)\,\|A\|_2. \end{aligned}$$

The assertion follows from Lemma 2.16. □

4.5.3 Numerical Results

The results shown in Table 4.23 were obtained for a finite element discretization of the operator $\mathscr{L} := -\mathrm{div}\, C(x)\nabla$ with random coefficients on the computational domain shown in Fig. 4.12. The computations were done on an Intel Xeon 3.0 GHz with 16 GB of core memory. The times required to compute the matrix partition based on the algebraic admissibility condition (4.50) and the geometric admissibility condition (4.11) scale almost linearly with the number of degrees of freedom. The accuracy of the approximate Cholesky factorization was chosen $\varepsilon_{\mathscr{H}} = 0.1$. Compared with the usual geometric approach to matrix partitioning, the algebraic method leads to a significantly faster computation of the approximate factorization, which is used for preconditioning. Additionally, the memory consumption of the algebraic $\mathscr{H}$-Cholesky factorization is reduced compared with the geometric method. In Table 4.23 we compared the results with the direct solver PARDISO[1] [230]. The core memory of 16 GB did not suffice for the largest example.

Depending on the needs of the user, it is possible to raise or decrease the accuracy $\varepsilon_{\mathscr{H}}$ of the algebraic $\mathscr{H}$-Cholesky factorization, which influences both the preconditioning effect and the cost of the preconditioner.

Table 4.23 $\mathscr{H}$-matrix preconditioned CG vs. direct solution.

	algebraic partitioning				geometric partitioning				PARDISO		
	setup		PCG		setup		PCG		setup		solver
size	time	MB	#It	time	time	MB	#It	time	time	MB	time
32 429	2s	15	8	0s	4s	22	6	0s	2s	41	0s
101 296	8s	48	10	1s	17s	85	8	1s	7s	209	1s
271 272	26s	157	13	4s	59s	270	11	5s	31s	775	1s
658 609	78s	437	13	10s	177s	725	13	16s	166s	2 642	2s
906 882	110s	552	18	20s	282s	1 073	16	29s	297s	4 047	3s
2 538 854	392s	1 840	23	77s	1 197s	3 400	20	112s	–	–	–

4.5.4 Parallel Approximate *LU* Factorization

Assume that $p = 2^L$, $L \in \mathbb{N}$, processors are at our disposal. It is well-known that in addition to reducing the fill-in, nested dissection reorderings have the advantage that the *LU* factorization allows for an efficient parallelization. This follows from the special structure of the matrix.

We present a parallel *LU* factorization as a recursion over ℓ. Consider a diagonal block A from the ℓth level, $\ell < L$, of the block cluster tree. The *LU* factorization of A can be computed from

[1] as contained in Intel Math Kernel Library 9.1.023

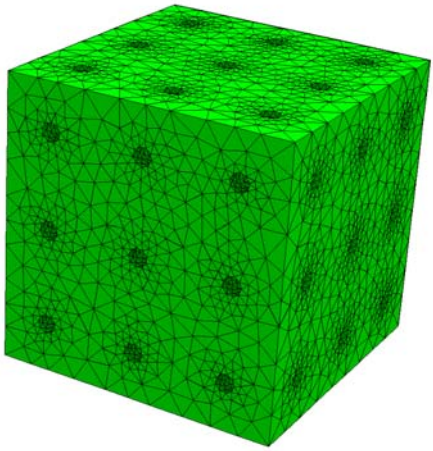


Fig. 4.12 Computational domain.

$$\begin{bmatrix} A_{11} & & A_{13} \\ & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} L_{11} & & \\ & L_{22} & \\ L_{31} & L_{32} & L_{33} \end{bmatrix} \begin{bmatrix} U_{11} & & U_{13} \\ & U_{22} & U_{23} \\ & & U_{33} \end{bmatrix}.$$

Indices 1 and 2 correspond to clusters separated by an interface. The latter corresponds to the index 3. The previous factorization is equivalent to computing

$$L_{11}, U_{11} \text{ from } A_{11} = L_{11}U_{11}, \qquad L_{22}, U_{22} \text{ from } A_{22} = L_{22}U_{22}, \tag{4.54a}$$

$$U_{13} \text{ from } A_{13} = L_{11}U_{13}, \qquad U_{23} \text{ from } A_{23} = L_{22}U_{23}, \tag{4.54b}$$

$$L_{31} \text{ from } A_{31} = L_{31}U_{11}, \qquad L_{32} \text{ from } A_{32} = L_{32}U_{22}, \tag{4.54c}$$

$$X_1 := L_{31}U_{13}, \qquad X_2 := L_{32}U_{23}, \tag{4.54d}$$

and

$$L_{33}, U_{33} \text{ from } L_{33}U_{33} = A_{33} - X_1 - X_2. \tag{4.55}$$

Hence, the two tasks in (4.54a), (4.54b), (4.54c), and (4.54d) can be done in parallel.

Since in the ℓth level $2^{L-\ell}$ processors can be used to compute the factorization, we use $2^{L-\ell-1}$ processors to solve the left part of (4.54) (the part corresponding to the first cluster) and the other $2^{L-\ell-1}$ processors to solve the right part (the second cluster). If $\ell = L$, then the sequential $\mathscr{H}$-matrix factorization algorithm from Sect. 2.9 is used.

Problem (4.54b) is a block forward substitution, i.e., a problem of type

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ A_{31} & A_{32} \end{bmatrix} = \begin{bmatrix} L_{11} & & \\ & L_{22} & \\ L_{31} & L_{32} & L_{33} \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \\ X_{31} & X_{32} \end{bmatrix} \tag{4.56}$$

has to be solved for X_{ij}, $i = 1,2,3$, $j = 1,2$. Assume this problem is to be solved with p processors. Then $p/2$ processors will be used to solve the left and the other $p/2$ processors will solve the right column of the following block forward substitutions

$$\begin{aligned} A_{11} &= L_{11}X_{11}, & A_{21} &= L_{22}X_{21}, \\ A_{12} &= L_{11}X_{12}, & A_{22} &= L_{22}X_{22}, \end{aligned}$$

and multiplications

$$\begin{aligned} Y_1 &:= L_{31}X_{11}, & Y_2 &:= L_{32}X_{21}, \\ Z_1 &:= L_{31}X_{12}, & Z_2 &:= L_{32}X_{22}. \end{aligned}$$

To parallelize the latter operations on p processors one can exploit the substructure

$$C := [A_1, A_2, A_3]^T [B_1, B_2, B_3] = A_1B_1 + A_2B_2 + A_3B_3,$$

which can be accomplished by computing each of the products

$$C_1 := A_1B_1, \quad C_2 := A_2B_2,$$

on $p/2$ processors and computing $C = A_3B_3 + C_1 + C_2$ on one of the p processors.

The last step to solve (4.56) is the computation of X_{31} and X_{32} from

$$L_{33}X_{31} = A_{31} - Y_1 - Y_2 \quad \text{and} \quad L_{33}X_{32} = A_{32} - Z_1 - Z_2.$$

Since L_{33} does not possess a nested dissection substructure, we solve each of the two problems sequentially on the first processor of each half of the p processors. To this end, Z_1 and Y_2 have to be sent to the respective processor, while Z_2 and Y_1 are already in the right place. Again, the case $\ell = L$ is treated by applying the sequential $\mathscr{H}$-matrix block forward substitution presented in Sect. 2.9. The backward substitution (4.54c) can be done analogously.

Only the last step (4.55), the computation of L_{33} and U_{33}, requires the completion of all previous steps (4.54a) through (4.54d). We solve (4.55) sequentially, which requires to communicate either X_1 or X_2. Since the dimension of (4.55) is significantly smaller than the previous problems, one can expect a reasonable speedup. The resulting distribution of blocks (and hence the memory distribution) among the processors is shown in Fig. 4.13 for $p = 4$.

4.5.4.1 Numerical Experiments

We test the parallel hierarchical *LU* factorization on the following magnetostatic problem

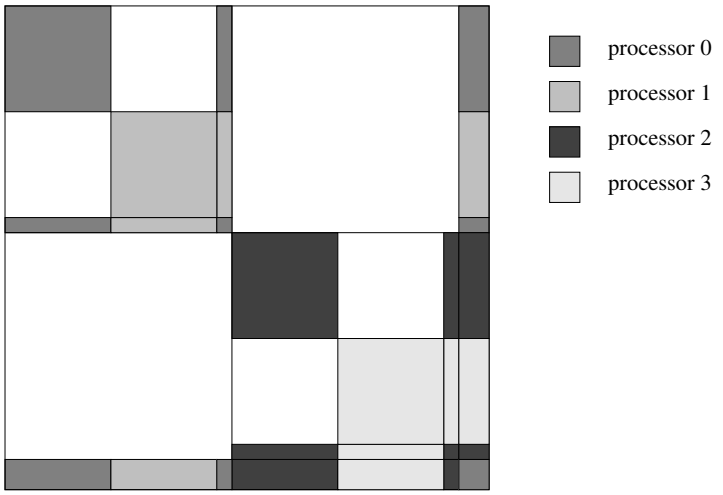


Fig. 4.13 Distribution of blocks among processors.

$$\operatorname{curl} \frac{1}{\mu} \operatorname{curl} A = j_0 \text{ in } \Omega, \tag{4.58a}$$

$$\operatorname{curl}_{\partial\Omega}(A \times \nu) = 0, \quad A \cdot \nu = 0 \text{ on } \Gamma_i, \tag{4.58b}$$

$$A \times \nu = 0 \text{ on } \Gamma_c. \tag{4.58c}$$

Here, $\operatorname{curl}_{\partial\Omega}$ denotes the surface vector curl and j_0 is a given source current. The bounded domain $\Omega = \Omega_C \cup \Omega_I \subset \mathbb{R}^3$ (see Fig. 4.15) is of simple topology and consists of different materials, which are characterized by their magnetic permeability $\mu := 4\pi 10^{-6}\mu_r$. The computational domain consists of a coil with material parameter $\mu_r = 1$ and a highly permeable core with $\mu_r = 500$, which are surrounded by air with $\mu_r = 1$. Therefore the magnetic permeability jumps between a value of $1.3 \cdot 10^{-6}$ in the air and the coil to a value of $6.5 \cdot 10^{-4}$ in the core. The contacts are denoted by $\Gamma_c := \partial\Omega \cap \partial\Omega_C$, and the rest of the boundary is given by $\Gamma_i := \partial\Omega \cap \partial\Omega_I$.

The magnetostatic problem (4.58) is singular. The magnetic field $B := \operatorname{curl} A$, which is the measurable physical quantity of interest, is not affected by an alternative solution $A_{\text{new}} = A + \nabla\phi$. An obvious idea to regularize the "magnetostatic operator" is adding a multiple of the identity. Hence, we consider the operator

$$\mathscr{L} := \operatorname{curl} \frac{1}{\mu} \operatorname{curl} + \alpha \mathscr{I} \tag{4.59}$$

with a constant $0 < \alpha \in \mathbb{R}$ as a regularization of the magnetostatic operator

$$\mathscr{L}_0 := \operatorname{curl} \frac{1}{\mu} \operatorname{curl}.$$

In the following tests we compute an approximate nested dissection Cholesky decomposition of edge element discretizations (see [146]) of the operator $\mathscr{L}$ in (4.59) with $\alpha = 500 \cdot 10^6$. This approximate Cholesky decomposition is used for preconditioning discretizations of $\mathscr{L}_0$ in the conjugate gradient method. For the spectral properties of the preconditioned systems the reader is referred to [31].

The results shown in Table 4.24 were obtained on a system consisting of four Intel Xeon 3.0 GHz processors. The hierarchical matrix rounding accuracy was chosen $\varepsilon_{\mathscr{H}} = 1_{10}{-}2$ in all tests, and the relative residual error of CG was set to $1_{10}{-}4$. Apparently, the complexity scales almost linearly (see Fig. 4.14) and the parallelization of the hierarchical Cholesky factorization algorithm shows a competitive speedup. The number of iterations is bounded independently of n.

Figure 4.15 presents the computed magnetostatic field. The upper left picture shows the setting, the upper right picture shows the current density j, and the lower pictures show the resulting magnetic field $B = \operatorname{curl} A$ on two different scales.

Table 4.24 Results obtained on $p = 1, 2, 4$ processors.

		Cholesky fact. time			
n	partition	$p=1$	$p=2$	$p=4$	#It
163 693	5.8s	39.9s	20.1s	12.7s	60
297 302	17.7s	88.0s	45.2s	24.6s	75
420 881	30.7s	151.6s	80.9s	46.9s	79
523 989	40.5s	182.5s	91.8s	56.6s	92
664 539	64.9s	266.9s	136.0s	75.5s	89
742 470	68.3s	310.0s	175.8s	93.2s	79
810 412	76.8s	340.9s	198.5s	108.1s	84
955 968	89.0s	399.3s	242.6s	126.3s	36

4.6 Solving Nonlinear Problems with Broyden Updates

The purpose of this section is the efficient numerical solution of second-order nonlinear elliptic Dirichlet problems

$$-\nabla \cdot [\alpha(u)\nabla u - \beta(u)u] + \gamma(u)u = f \quad \text{in } \Omega, \tag{4.60a}$$

$$u = g \quad \text{on } \partial\Omega, \tag{4.60b}$$

where Ω is a bounded Lipschitz domain in $\mathbb{R}^d$ and f, g are given. The coefficient functions α, β, and γ are assumed to be in $W^{1,\infty}(\Omega)$. In addition, for the coefficient α appearing in the principal part there is a real number $\alpha_0 > 0$ such that $\alpha(u) \geq \alpha_0$.

Nonlinear elliptic problems arise from many applications in physics and other fields, for instance in elastoplasticity, magnetic potential problems, viscous fluid flow problems, chemical reactions, and pattern formation in biology. During the past three decades, many numerical approaches have been developed for solving

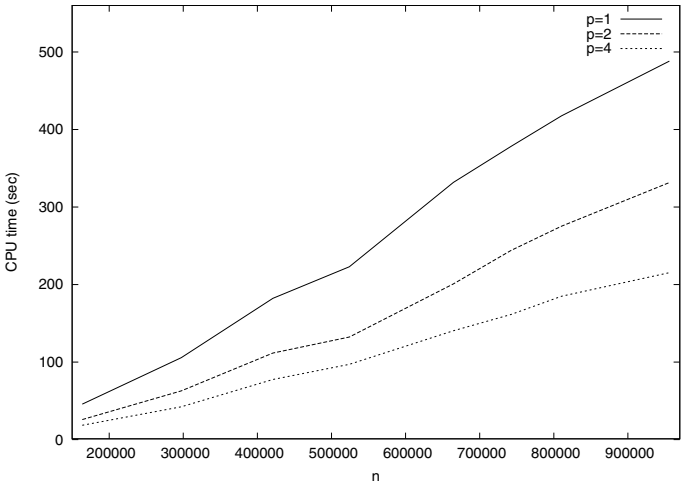


Fig. 4.14 Preconditioner setup time for $p = 1,2,4$ processors.

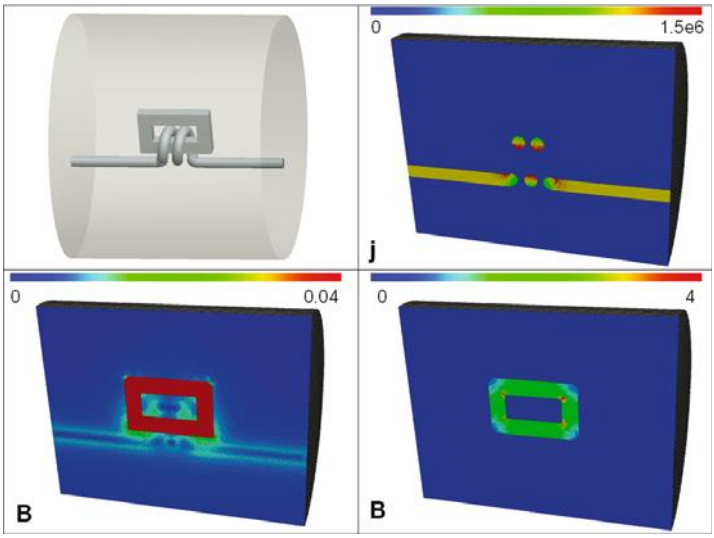


Fig. 4.15 Results of the magnetostatic field computations.

nonlinear problems; see, e.g., [8, 64]. Undoubtedly, **Newton's method** (see [155, 82])

$$u_{k+1} = u_k - (F'(u_k))^{-1} F(u_k), \quad k = 0,1,2,\ldots, \tag{4.61}$$

for the solution of the nonlinear problem $F(u) = 0$ is one of the most popular methods, which receives attention from applications due to its quadratic convergence.

Newton's method approximates the nonlinear problem (4.60) or equivalently

$$F(u) := -\nabla \cdot [\alpha(u)\nabla u - \beta(u)u] + \gamma(u)u - f = 0 \tag{4.62}$$

by a sequence of linear problems

$$F'(u_k)(u_{k+1} - u_k) = -F(u_k), \quad k = 0, 1, 2, \ldots. \tag{4.63}$$

Note that since

$$F'(u)v = -\nabla \cdot \big[\alpha(u)\nabla v - \{\beta(u) + \beta'(u)u - \alpha'(u)\nabla u\}v\big] + \{\gamma(u) + \gamma'(u)u\}v,$$

the assumption $\alpha(u) \geq \alpha_0 > 0$ guarantees that each linear problem is in the class of second-order elliptic Dirichlet boundary value problems (4.1).

Newton's method can be used to treat nonlinear elliptic equations discretized not only by the finite element method (see [150]) but also by the finite volume method; see [64]. From the Galerkin method applied to (4.63) we obtain in each Newton step a linear algebraic system

$$A(\underline{u}_k)\underline{\delta}_k = -F(\underline{u}_k) \tag{4.64}$$

for the unknown vector

$$\underline{\delta}_k := \underline{u}_{k+1} - \underline{u}_k.$$

Here, $\underline{u}_k \in \mathbb{R}^n$ defines the Galerkin approximation $\sum_{i=1}^n (\underline{u}_k)_i \varphi_i$ of u_k. Starting from a given vector $\underline{u}_0 \in \mathbb{R}^n$, $\underline{u}_k$ is updated by setting $\underline{u}_{k+1} = \underline{u}_k + \underline{\delta}_k$ until some given tolerance is reached. For the sake of readability in the following we will not distinct between the vector $\underline{u}_k \in \mathbb{R}^n$ and u_k from (4.63).

The update of u_k requires the solution of (4.64). Since its exact solution is expensive for large n, it is advisable to replace the coefficient matrix $A(u_k)$ by an approximation which allows a more efficient solution. This class of variants of Newton's method is called **quasi-Newton methods**. Avoiding the computation of the Jacobian and its inverse in each Newton step, instead of (4.61) the iteration reads

$$u_{k+1} = u_k - B_k F(u_k), \quad k = 0, 1, 2, \ldots,$$

where B_k denotes a matrix which approximates the inverse $(F'(u_k))^{-1}$ of the Jacobian. A general description of quasi-Newton methods can be found in [80, 13, 81, 82, 155] and its applications to elliptic problems in [8, 153]. Another class are **inexact Newton methods** [76], which compute the vectors $(F'(u_k))^{-1}F(u_k)$, $k = 0, 1, 2, \ldots$, usually by a Krylov subspace method. This approach, however, requires a robust and efficient preconditioner.

Since $F'(u_k)$ is an elliptic second order of type (4.2), due to Theorem 4.28 it is possible to compute an approximate inverse of $A(u_k)$ with almost linear complexity. Hence, we can easily accelerate Newton's method by employing $\mathscr{H}$-matrices.

4.6.1 Broyden Updates

A special quasi-Newton method is **Broyden's method** [53], which starts from a given approximation $A_0 \in \mathbb{R}^{n\times n}$ of $A(u_0)$ and computes approximations A_k of $A(u_k)$ using rank-1 updates

$$A_{k+1} = A_k + \frac{(y_k - A_k\delta_k)\delta_k^T}{\delta_k^T\delta_k}, \tag{4.65}$$

where $y_k := F(u_{k+1}) - F(u_k)$. The matrix A_{k+1} can be seen to be the closest matrix B to A_k satisfying the **secant equation** $B\delta_k = y_k$; i.e., with $\mathcal{Q} := \{B \in \mathbb{R}^{n\times n} : B\delta_k = y_k\}$ it holds that

$$\|A_{k+1} - A_k\|_2 = \min_{B\in\mathcal{Q}} \|B - A_k\|_2.$$

As for other quasi-Newton methods, the quadratic convergence of Newton's method is reduced to a super-linear one; cf. [81]. Hence, usually more steps will be required to obtain the same accuracy as Newton's method. Despite this fact, Broyden's method is significantly cheaper than Newton's method. A disadvantage of Broyden's method is that the whole history of update vectors has to be stored. If many iteration steps are required, then this will seriously affect the required amount of storage.

Since one is interested in the solution of (4.64), it is crucial that the inverse $B_k := A_k^{-1}$ can be updated in a similar way as A_k is updated in (4.65). With the Sherman-Morrison formula (1.3) we have that

$$A_{k+1}^{-1} = A_k^{-1} + \frac{(\delta_k - A_k^{-1}y_k)\delta_k^T A_k^{-1}}{\delta_k^T A_k^{-1}\delta_k}. \tag{4.66}$$

Hence, if $B_0 = A_0^{-1}$ has been found, B_{k+1} can be computed from B_k by

$$B_{k+1} = B_k + \frac{(\delta_k - B_k y_k)\delta_k^T B_k}{\delta_k^T B_k y_k} = B_k + \frac{c_k z_k^T}{\|\delta_k\|_2^2 - \delta_k^T c_k},$$

where we have exploited that $B_k y_k = B_k F(u_{k+1}) - B_k F(u_k) = \delta_k - c_k$ with

$$c_k := -B_k F(u_{k+1}) \quad \text{and} \quad z_k := B_k^T\delta_k.$$

Furthermore, using a trick from [83],

$$\begin{aligned}\delta_{k+1} = -B_{k+1}F(u_{k+1}) &= -\left(I + \frac{c_k\delta_k^T}{\|\delta_k\|_2^2 - \delta_k^T c_k}\right) B_k F(u_{k+1}) \\ &= \left(I + \frac{c_k\delta_k^T}{\|\delta_k\|_2^2 - \delta_k^T c_k}\right) c_k = \left(1 + \frac{\delta_k^T c_k}{\|\delta_k\|_2^2 - \delta_k^T c_k}\right) c_k.\end{aligned}$$

Hence, c_k and δ_{k+1} are linearly dependent. We obtain Algorithm 4.1, which involves only two matrix-vector multiplications and a rank-1 update per iteration step.

Let u_0 be given; Compute $B_0 = A(u_0)^{-1}$.
$\delta_0 = -B_0 F(u_0)$
for $k = 0, 1, 2, \ldots$ **do**
$u_{k+1} = u_k + \delta_k$
$c_k = -B_k F(u_{k+1})$
$r := \delta_k^T c_k$, $s := \|\delta_k\|_2^2 - r$
$z_k = \frac{1}{s} B_k^T \delta_k$
$B_{k+1} = B_k + c_k z_k^T$
$\delta_{k+1} = \left(1 + \frac{r}{s}\right) c_k$

Algorithm 4.1: Broyden's method.

Compared with Newton's method, only one inversion has to be computed when the initial matrix B_0 is generated.

Multiplying a matrix by a vector and adding a rank-1 matrix with fixed yet arbitrary precision can be performed in the set of $\mathscr{H}$-matrices with almost linear complexity. Hence, we obtain another obvious but efficient variant of Newton's method. Note that in this approach it is not necessary to store the update vectors separately. We are able to gradually add the updates to B_0, if we form B_0 instead of only applying it to a vector. It is known (cf. [81]) that B_k is an approximation of the inverse $(F'(u_k))^{-1}$ of the Jacobian. Hence, the complexity of B_{k+1} can be expected to be of the same order as B_k; cf. Theorem 4.28. Compared with the $\mathscr{H}$-matrix accelerated Newton method only one $\mathscr{H}$-matrix inversion has to be computed when the initial matrix B_0 is generated. Matrix-vector multiplications and rank-1 updates can be expected to be done with much less time consumption than computing an approximate inverse in each step.

Remark 4.38. Since adding the rank-1 updates explicitly to the matrix requires some additional time, it may be worth gathering a constant number ν of updates before this rank-ν matrix is actually added to the approximation of the Jacobian.

According to Theorem 4.35, also the factors L and U in the LU decomposition of the discretization of the Jacobian can be approximated with almost linear complexity. As we have seen, the computation of these approximations can be done in significantly less time than the computation of the hierarchical inverse while the same robustness with respect to varying coefficients of the operator can be observed. Hence, it seems desirable to replace the role of the inverse in (4.66) by the LU decomposition. However, no update formula like the Sherman-Morrison formula is known for the factors of the LU decomposition. In the following paragraph we present a method which can be used for updating the factors L and U of an LU decomposition of A whenever A is updated by a rank-1 matrix.

4.6.2 An Update Method for the LU Decomposition

Let A be decomposed using the LU decomposition $A = LU$. Assume we want to update the matrix A with a given rank-ν matrix XY^H, $X, Y \in \mathbb{R}^{n \times \nu}$, such that

$A+XY^H$ is still non-singular. Then the triangular factors of $A+XY^H$ can be computed from an LU decomposition of $I+AB^H = L'U'$, where $LA = X$ and $U^HB = Y$. This can be seen from

$$A+XY^H = L(I+L^{-1}XY^HU^{-1})U = L(I+AB^H)U = (LL')(U'U)$$

and the fact that the product of unit lower and upper triangular matrices is unit lower and upper triangular, respectively.

The following lemma (see [24]) shows that the LU decomposition of low-rank updates of the identity leads to diagonal-plus-semi-separable triangular factors; cf. Sect. 1.2.

Lemma 4.39. *Let $A,B \in \mathbb{C}^{n\times\nu}$ and assume that $I+AB^H$ has the LU decomposition*

$$I+AB^H = L'U'$$

with a unit lower triangular matrix L' and an upper triangular matrix U'. Then L' and U' are diagonal-plus-semi-separable, i.e., there are matrices $S_\ell \in \mathbb{C}^{\nu\times\nu}$, $\ell = 1,\ldots,n$, defined by $S_1 = I$ and

$$S_{\ell+1} = (I - \frac{S_\ell b_\ell a_\ell^H}{1+a_\ell^H S_\ell b_\ell})S_\ell, \quad \ell = 1,\ldots,n-1,$$

such that for $\ell = 1,\ldots,n$ it holds that $U'_{\ell\ell} = 1+a_\ell^H S_\ell b_\ell$,

$$L'_{i\ell} = \frac{1}{U'_{\ell\ell}} a_i^H S_\ell b_\ell, \quad U'_{\ell i} = a_\ell^H S_\ell b_i, \quad i > \ell. \tag{4.67}$$

Here, $a_\ell, b_\ell \in \mathbb{C}^\nu$ denote the ℓth row of A and B, respectively.

Proof. The assertion is proved by induction over ℓ. For the first column and the first row of $L'U'$ one has $U'_{11} = (L'U')_{11} = 1+a_1^H b_1$ and for $i > 1$

$$L'_{i1}U'_{11} = (L'U')_{i1} = a_i^H b_1, \quad U'_{1i} = (L'U')_{1i} = a_1^H b_i.$$

Assume that (4.67) is valid for $\ell-1$, then the following entries appear in the ℓth column and the ℓth row of $L'U'$

$$U'_{\ell\ell} = (L'U')_{\ell\ell} - \sum_{j=1}^{\ell-1} L'_{\ell j}U'_{j\ell} = 1+a_\ell^H \left(I - \sum_{j=1}^{\ell-1} \frac{1}{U'_{jj}} S_j b_j a_j^H S_j\right) b_\ell,$$

$$L'_{i\ell}U'_{\ell\ell} = (L'U')_{i\ell} - \sum_{j=1}^{\ell-1} L'_{ij}U'_{j\ell} = a_i^H \left(I - \sum_{j=1}^{\ell-1} \frac{1}{U'_{jj}} S_j b_j a_j^H S_j\right) b_\ell, \quad i > \ell,$$

and

$$U'_{\ell i} = (L'U')_{\ell i} - \sum_{j=1}^{\ell-1} L'_{\ell j}U'_{ji} = a_\ell^H \left(I - \sum_{j=1}^{\ell-1} \frac{1}{U'_{jj}} S_j b_j a_j^H S_j\right) b_i, \quad i > \ell.$$

Setting $S_\ell = I - \sum_{j=1}^{\ell-1} \frac{1}{U'_{jj}} S_j b_j a_j^H S_j$, we obtain

$$S_{\ell+1} = S_\ell - \frac{1}{U'_{\ell\ell}} S_\ell b_\ell a_\ell^H S_\ell = (I - \frac{1}{U'_{\ell\ell}} S_\ell b_\ell a_\ell^H) S_\ell = (I - \frac{S_\ell b_\ell a_\ell^H}{1 + a_\ell^H S_\ell b_\ell}) S_\ell$$

and hence the assertion. □

For the computation of the factors L' and U' it is sufficient to compute the matrices $S_\ell \in \mathbb{C}^{\nu\times\nu}$ and $U'_{\ell\ell}$, $\ell = 1,\dots,n$, which requires $(\nu^2+1)n$ units of storage and $\mathcal{O}(n)$ arithmetic operations.

Instead of storing the matrices L' and U' for each rank-ν update AB^H, we account for them by explicitly updating the factors L and U. Hence, we devise an explicit update procedure for the factors of the LU decomposition. Exploiting the semi-separable structure of L' and U', the products LL' and UU' can be computed with small effort as described in Sect. 2.7.4. Hence, the update of the LU decomposition $A = LU$ with XY^H can be done by the following three steps

(i) Solve $LA = X$ and $U^H B = Y$ for $A, B \in \mathbb{C}^{n\times\nu}$;
(ii) Compute S_ℓ and $U'_{\ell\ell}$, $\ell = 1,\dots,n$, as described in Lemma 4.39;
(iii) Update L and U by multiplying L by L' from the right and U by U' from the left; see Algorithms 2.6 and 2.7.

It is obvious how to modify the previous algorithms if the Cholesky decomposition of a symmetric positive definite matrix A is to be updated by a rank-ν matrix XX^H.

With these ideas, a variant of Broyden's method which is based on the LU decomposition can be constructed. In the rest of this section this method will be referred to as the *$\mathcal{H}$-Broyden-LU method*. We have seen in Sect. 2.7.4 that the third step in the preceding algorithm can be done with complexity $\mathcal{O}(r^2 n \log n)$ if the updates are explicitly added to an $\mathcal{H}(P,r)$-matrix; see part (b) of Remark 2.32.

The complexity of the hierarchical forward/backward substitution required in (i) is of the same order as the hierarchical matrix-vector multiplication; cf. [23]. The initial hierarchical LU decomposition of A_0 requires $r^2 n(\log n)^2$ operations. Hence, the number of arithmetic operations for k steps of this variant (see Algorithm 4.2) of Broyden's method in $\mathcal{H}(P,r)$ is of the order $r^2 n \log n(k + \log n)$.

Let u_0 be given; Factorize $A(u_0) = L_0 U_0$.
Set $b := F(u_0)$.
for $k = 0,1,2,\dots$ **do**
  solve $L_k U_k \delta_k = -b$ for δ_k
  $u_{k+1} = u_k + \delta_k$
  set $b = F(u_{k+1})$
  solve $L_k s_k = b$ for s_k
  solve $U_k^T z_k = \delta_k / \|\delta_k\|_2^2$ for z_k
  update $L_{k+1} U_{k+1} = L_k (I + s_k z_k^T) U_k$.

Algorithm 4.2: Broyden's method based on triangular update.

The gathered version (see Remark 4.38) of Algorithm 4.2 additionally requires the forward/backward solution with triangular semi-separable matrices. The forward

problem $Lx = b$ (and similarly the backward problem) can be solved with linear complexity $\mathcal{O}(n)$ due to $x_1 = b_1$ and

$$x_i = b_i - w_{i-1} \sum_{j=1}^{i-1} z_j x_j, \quad i = 2, \ldots, n.$$

4.6.3 The Influence of Truncation Errors

The truncation of the sum of low-rank matrices introduces an additional error matrix E_k, i.e., instead of the Broyden update (4.65) we have $A_{k+1} = A'_{k+1} + E_{k+1}$, where

$$A'_{k+1} = A_k + \frac{(y_k - A_k \delta_k)\delta_k^T}{\delta_k^T \delta_k}.$$

In this section we will prove super-linear convergence of this perturbed method if we assume that

$$\|E_k\| \leq c\|\delta_k\|, \tag{4.68}$$

where c does not depend on k. This can be done by adaptation of the proofs presented in [81].

In [81, Thm. 8.2.2] it is shown that the local convergence of Broyden's method follows from the so-called **bounded-deterioration property**

$$\|A'_{k+1} - J(x)\| \leq \|A_k - J(x)\| + \frac{\gamma}{2}(\|e_{k+1}\| + \|e_k\|), \tag{4.69}$$

where $e_k := u_k - u$ and γ is the Lipschitz constant for F' at u. In our case we need this property for A_{k+1} rather than for A'_{k+1}. By the triangle inequality and (4.68) it follows that

$$\|A_{k+1} - J(u)\| \leq \|A'_{k+1} - J(u)\| + c\|\delta_k\| \leq \|A_k - J(u)\| + (\frac{\gamma}{2} + c)(\|e_k\| + \|e_{k+1}\|)$$

due to $\|\delta_k\| \leq \|e_k\| + \|e_{k+1}\|$. Hence, we obtain property (4.69) with a larger constant and hence the local convergence.

A consequence of the local convergence is that

$$\|e_{k+1}\| \leq \|e_k\|/2. \tag{4.70}$$

The following arguments show that the convergence is actually super-linear. Let $F_k = A_k - J(u)$. According to [81, Thm. 8.2.4] a sufficient condition for $\{u_k\}$ to converge super-linearly is

$$\lim_{k\to\infty} \frac{\|F_k \delta_k\|}{\|\delta_k\|} = 0. \tag{4.71}$$

Just as in [81] one can derive the following improved property of bounded deterioration

$$\|A'_{k+1} - J(u)\|_F \leq \|F_k\|_F - \frac{\|F_k\delta_k\|^2}{2\|F_k\|_F\|\delta_k\|^2} + \frac{3}{4}\gamma\|e_k\|,$$

which proves that

$$\begin{aligned}\|F_{k+1}\|_F &\leq \|F_k\|_F - \frac{\|F_k\delta_k\|^2}{2\|F_k\|_F\|\delta_k\|^2} + \frac{3}{4}\gamma\|e_k\| + \|E_k\|_F \\ &\leq \|F_k\|_F - \frac{\|F_k\delta_k\|^2}{2\|F_k\|_F\|\delta_k\|^2} + \frac{3}{4}(\gamma+2c)\|e_k\|.\end{aligned}$$

The previous estimate follows from (4.68) and (4.70), and it is equivalent to

$$\frac{\|F_k\delta_k\|^2}{\|\delta_k\|^2} \leq 2\|F_k\|_F\left(\|F_k\|_F - \|F_{k+1}\|_F + \frac{3}{4}(\gamma+2c)\|e_k\|\right). \tag{4.72}$$

Since $\{F_k\}_k$ is bounded, i.e., there is $\bar{c} > 0$ such that $\|F_k\|_F \leq \bar{c}$ for all $k \in \mathbb{N}$, and since (4.70) implies that

$$\sum_{k=0}^{\infty}\|e_k\| \leq 2\|e_0\|,$$

summing up (4.72) gives

$$\sum_{k=0}^{\infty}\frac{\|F_k\delta_k\|^2}{\|\delta_k\|^2} \leq 2\bar{c}\left(\|F_0\|_F + \frac{3}{2}(\gamma+2c)\|e_0\|\right) \leq 2\bar{c}\left(\|F_0\| + \frac{3}{2}(\gamma+2c)\|e_0\|\right).$$

Hence, (4.71) is satisfied and the super-linear convergence is proved.

4.6.4 Numerical Experiments

In this section we report results of numerical experiments which confirm the efficiency of the $\mathscr{H}$-Broyden-*LU* method when solving nonlinear elliptic problems. It will be seen that the complexity of the latter method scales almost linearly with respect to the number of degrees of freedom n. We compare this method with two standard Broyden methods. One is based on the usual *LU* factorization instead of the inverse. For the factorization we employ the PARDISO library [230] as part of the Intel Math Kernel Library version 9.1. The columns containing the corresponding results will be labeled "Broyden-PARDISO". For the second standard Broyden method we apply Boomer-AMG from the HYPRE software library [1] instead of the inverse. The corresponding results will be labeled "Broyden-AMG".

The experiments were carried out on a single processor of an Intel Core2 Duo 3.0 GHz with 16 GB of core memory. For all tests we use piecewise linear basis

functions. In the following we list two examples of the model problem (4.60). The first is in two spatial dimensions, the second is three dimensional.

Numerical results are listed in Tables 4.25 and 4.26. "#It" denotes the number of iterations, "MB" stands for the memory requirement of the respective method in MBytes. "time" consists of two parts: The first value is the time in seconds required for computing the *LU* decomposition and for setting up AMG, respectively, and the second value is for Broyden's iteration. The iteration was stopped if the difference of two consecutive approximations u_k and u_{k+1} satisfies $\|u_{k+1} - u_k\|_2 < 10^{-6}$.

Example 4.40. We consider the stationary *viscous Burgers' equation*

$$-\Delta u + u(u_x + u_y) = f \quad \text{in } \Omega = (0,1)^2,$$

i.e., we set $A(u) = 1$, $b(u) = \frac{1}{2}(u,u)^T$, and $c(u) = 0$ in (4.60). To obtain the reference solution

$$u = 50\cos(2\pi x + \frac{\pi}{2})(y^2 - y^3),$$

we use the right-hand side

$$f = 100\cos(2\pi x + \frac{\pi}{2})\Big((y^2 - y^3)[2\pi^2 + 25\cos(2\pi x + \frac{\pi}{2})(2y - 3y^2) - 50\pi\sin(2\pi x + \frac{\pi}{2})(y^2 - y^3)] + 3y - 1\Big).$$

Table 4.25 Iterations and time for Example 4.40.

	Broyden-PARDISO				Broyden-AMG				$\mathscr{H}$-Broyden-*LU*			
n	#It	time		MB	#It	time		MB	#It	time		MB
62 001	55	1s	22s	155	55	0s	38s	*52*	65	1s	42s	60
123 904	54	3s	46s	344	54	1s	82s	*103*	58	3s	85s	128
249 001	52	6s	89s	679	52	1s	161s	*198*	57	7s	182s	265
498 436	54	15s	198s	1425	54	2s	348s	*410*	74	16s	521s	565
998 001	54	34s	362s	2959	54	4s	711s	*821*	57	35s	821s	1125
1 996 569	53	83s	751s	6191	53	8s	1408s	*1611*	58	71s	1705s	2230

Table 4.25 shows that the times for computing the classical and the approximate *LU* factorization are comparable. Updating the matrix during the Broyden iteration and applying it takes 2–3 times more time than applying the updates in Broyden-PARDISO. One of the reasons is that a slightly larger number of iterations is required in $\mathscr{H}$-Broyden-*LU*. However, the memory consumption of Broyden-PARDISO (including the update vectors) is almost three times the memory used by $\mathscr{H}$-Broyden-*LU*. Broyden-AMG is almost two times slower than Broyden-PARDISO but slightly faster than $\mathscr{H}$-Broyden-*LU*. We were not able to find out the memory consumption of Boomer-AMG. Therefore, the column labeled "MB" contains only the memory required for storing the update vectors.

Example 4.41. As the second example we consider the so-called *nonlinear Schrödinger equation* with nonlinear diffusion

$$-\mathrm{div}(u+1)\nabla u + 2|u|^2 u = f \quad \text{in } \Omega := (0,1)^3.$$

Hence, in view of (4.60) we choose $A(u) = u+1$, $b(u) = 0$, $c(u) = 2|u|^2$. In order to validate the solution, we use the right-hand side f such that (4.60) has the exact solution

$$u = x(1-x)\sin(\pi y)\cos(\pi z + \frac{\pi}{2}).$$

Table 4.26 Iterations and time for Example 4.41.

	Broyden-PARDISO				Broyden-AMG				$\mathscr{H}$-Broyden-*LU*			
n	#It	time		MB	#It	time		MB	#It	time		MB
54 872	51	7s	28s	375	51	1s	73s	*42*	56	15s	54s	106
117 649	51	29s	69s	1039	50	2s	193s	*90*	65	28s	118s	251
238 328	49	104s	134s	2480	49	5s	410s	*167*	52	69s	211s	541
474 552	50	448s	350s	6783	50	10s	965s	*365*	64	157s	596s	1115
970 299	–	–	–	–	50	23s	2031s	*740*	74	378s	1523s	2543

In this example $\mathscr{H}$-Broyden-*LU* requires less time than Broyden-AMG although the number of iterations is slightly larger. The memory consumption of Broyden-AMG could not be determined. The value for Broyden-AMG in Table 4.26 is the memory required for the update vectors. While Broyden-PARDISO was the fastest method in Example 4.40 in two spatial dimensions, the asymptotic super-linear complexity of the classical *LU* factorization is revealed in this three-dimensional example. While the latter method is faster than $\mathscr{H}$-Broyden-*LU* for small n, it becomes less efficient for larger n. We were not able to compute the largest problem ($n = 970\,299$) since the method ran out of memory.

We want to remark that the fixed Newton method could not be used in our numerical experiments, since the iteration did not converge in any example.

References

1. Hypre – high performance preconditioners. `http://www.llnl.gov/CASC/hypre/`.
2. R. A. Adams. *Sobolev Spaces*. Academic Press, 1975.
3. B. K. Alpert, G. Beylkin, R. Coifman, and V. Rokhlin. Wavelet-like bases for the fast solution of second-kind integral equations. *SIAM J. Sci. Comput.*, 14:159–184, 1993.
4. P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM J. Matrix Anal. Appl.*, 23(1):15–41 (electronic), 2001.
5. S. Amini and A. Profit. Multi-level fast multipole solution of the scattering problem. *Engineering Analysis with Boundary Elements*, 27(5):547–654, 2003.
6. A. W. Appel. An efficient program for many-body simulation. *SIAM J. Sci. Statist. Comput.*, 6:85–103, 1985.
7. E. Asplund. Inverses of matrices $\{a_{ij}\}$ which satisfy $a_{ij} = 0$ for $j \geq i + p$. *Math. Scand.*, 7:57–60, 1959.
8. O. Axelsson. On global convergence of iterative methods. In *Iterative Solution of Nonlinear Systems of Equations*, volume 953 of *Lecture Notes in Math.*, pages 1–19. Springer, Berlin, 1982.
9. O. Axelsson and G. Lindskog. On the rate of convergence of the preconditioned conjugate gradient method. *Numer. Math.*, 48(5):499–523, 1986.
10. O. Axelsson and P. Vassilevski. Algebraic multilevel preconditioning methods I. *Numer. Math.*, 56:157–177, 1989.
11. O. Axelsson and P. Vassilevski. Algebraic multilevel preconditioning methods II. *SIAM J. Numer. Anal.*, 27:1569–1590, 1990.
12. L. Banjai and W. Hackbusch. Hierarchical matrix techniques for low- and high-frequency Helmholtz problems. *IMA Journal of Numerical Analysis*, 28:46–79, 2008.
13. R. E. Bank and D. J. Rose. Global approximate Newton methods. *Numer. Math.*, 37:279–295, 1981.
14. J. Barnes and P. Hut. A hierarchical $\mathcal{O}(n \ln n)$ force calculation algorithm. *Nature*, 324: 446–449, 1986.
15. W. Barrett and P. J. Feinsilver. Inverses of banded matrices. *Lin. Alg. Appl.*, 41:111–130, 1981.
16. B. J. C. Baxter and G. Roussos. A new error estimate of the fast Gauss transform. *SIAM J. Sci. Comput.*, 24(1):257–259 (electronic), 2002.
17. M. Bebendorf. Approximation of boundary element matrices. *Numer. Math.*, 86(4):565–589, 2000.
18. M. Bebendorf. *Effiziente numerische Lösung von Randintegralgleichungen unter Verwendung von Niedrigrang-Matrizen*. PhD thesis, Universität Saarbrücken, 2000. dissertation.de, Verlag im Internet, 2001. ISBN 3-89825-183-7.

19. M. Bebendorf. A note on the Poincaré inequality for convex domains. *J. Anal. Appl.*, 22: 751–756, 2003.
20. M. Bebendorf. Efficient inversion of Galerkin matrices of general second-order elliptic differential operators with nonsmooth coefficients. *Math. Comp.*, 74:1179–1199, 2005.
21. M. Bebendorf. Hierarchical *LU* decomposition based preconditioners for BEM. *Computing*, 74:225–247, 2005.
22. M. Bebendorf. Approximate Inverse Preconditioning of FE Systems for Elliptic Operators with Non-smooth Coefficients. *SIAM J. Matrix Anal. Appl.*, 27(4):909–929, 2006.
23. M. Bebendorf. Why finite element discretizations can be factored by triangular hierarchical matrices. *SIAM J. Num. Anal.*, 45(4):1472–1494, 2007.
24. M. Bebendorf and Y. Chen. Efficient solution of nonlinear elliptic problems using hierarchical matrices with broyden updates. *Computing*, 81:239–257, 2007.
25. M. Bebendorf and Th. Fischer. On the purely algebraic data-sparse approximation of the inverse and the triangular factors of sparse matrices. Technical report, 2008. submitted.
26. M. Bebendorf and R. Grzhibovskis. Accelerating Galerkin BEM for Linear Elasticity using Adaptive Cross Approximation. *Mathematical Methods in the Applied Sciences*, 29: 1721–1747, 2006.
27. M. Bebendorf and W. Hackbusch. Existence of $\mathscr{H}$-matrix approximants to the inverse FE-matrix of elliptic operators with L^∞-coefficients. *Numer. Math.*, 95(1):1–28, 2003.
28. M. Bebendorf and W. Hackbusch. Stabilised rounded addition of hierarchical matrices. *Num. Lin. Alg. Appl.*, 14(5):407–423, 2007.
29. M. Bebendorf and R. Kriemann. Fast parallel solution of boundary integral equations and related problems. *Comput. Visual. Sci.*, 8:121–135, 2005.
30. M. Bebendorf and S. Kunis. Recompression techniques for adaptive cross approximation. Technical report, Preprint INS Bonn, 2007. submitted.
31. M. Bebendorf and J. Ostrowski. Parallel hierarchical matrix preconditioners for the curl-curl operator. Technical report, 2008. submitted.
32. M. Bebendorf and S. Rjasanow. Adaptive low-rank approximation of collocation matrices. *Computing*, 70(1):1–24, 2003.
33. M. Bebendorf, S. Rjasanow, and E. E. Tyrtyshnikov. Approximation using diagonal-plus-skeleton matrices. In *Mathematical aspects of boundary element methods (Palaiseau, 1998)*, volume 414 of *Chapman & Hall/CRC Res. Notes Math.*, pages 45–52. Chapman & Hall/CRC, Boca Raton, FL, 2000.
34. I. Benedetti, M. H. Aliabadi, and G. Davi. A fast 3d dual boundary element method based on hierarchical matrices. *International Journal of Solids and Structures*, 45:2355–2376, 2008.
35. M. Benzi. Preconditioning techniques for large linear systems: a survey. *J. Comput. Phys.*, 182(2):418–477, 2002.
36. M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numer.*, 14:1–137, 2005.
37. M. Benzi and M. Tůma. A sparse approximate inverse preconditioner for nonsymmetric linear systems. *SIAM J. Sci. Comput.*, 19:968–994, 1998.
38. M. Benzi and M. Tůma. A comparative study of sparse approximative inverse preconditioners. *Appl. Numer. Math.*, 30:305, 1999.
39. M. Bern and D. Eppstein. Approximation algorithms for geometric problems. In D. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*. PWS Publishing Company, Boston, 1997.
40. G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms. I. *Comm. Pure Appl. Math.*, 44(2):141–183, 1991.
41. S. Börm. Approximation of integral operators by $\mathscr{H}^2$-matrices with adaptive bases. *Computing*, 74(3):249–271, 2005.
42. S. Börm. $\mathscr{H}^2$-matrix arithmetics in linear complexity. *Computing*, 77(1):1–28, 2006.
43. S. Börm and L. Grasedyck. Hybrid cross approximation of integral operators. *Numer. Math.*, 101(2):221–249, 2005.

44. S. Börm and W. Hackbusch. Approximation of boundary element operators by adaptive $\mathscr{H}^2$-matrices. In *Foundations of computational mathematics: Minneapolis, 2002*, volume 312 of *London Math. Soc. Lecture Note Ser.*, pages 58–75. Cambridge Univ. Press, Cambridge, 2004.
45. H. Brakhage and P. Werner. Über das Dirichletsche Außenraumproblem für die Helmholtzsche Schwingungsgleichung. *Arch. Math.*, 16:325–329, 1965.
46. J. H. Bramble. *Multigrid methods*. Longman Scientific & Technical, 1993.
47. J. H. Bramble, Z. Leyk, and J. E. Pasciak. The analysis of multigrid algorithms for pseudodifferential operators of order minus one. *Math. Comp.*, 63(208):461–478, 1994.
48. J. H. Bramble, J. E. Pasciak, and J. Xu. Parallel multilevel preconditioners. *Math. Comp.*, 55(191):1–22, 1990.
49. A. Brandt. Multilevel computations of integral transforms and particle interactions with oscillatory kernels. *Comput. Phys. Comm.*, 65:24–38, 1991.
50. A. Brandt and A. A. Lubrecht. Multilevel matrix multiplication and fast solution of integral equations. *J. Comput. Phys.*, 90(2):348–370, 1990.
51. A. Brandt and C. H. Venner. Multilevel evaluation of integral transforms with asymptotically smooth kernels. *SIAM J. Sci. Comput.*, 19(2):468–492 (electronic), 1998.
52. J. Breuer. *Schnelle Randelementmethoden zur Simulation von elektromagnetischen Wirbelstromfeldern sowie ihrer Wärmeproduktion und Kühlung*. PhD thesis, Universität Stuttgart, 2005.
53. C. G. Broyden. A class of methods for solving nonlinear simultaneous equations. *Math. Comp.*, 19:577–593, 1965.
54. H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.
55. A. J. Burton and G. F. Miller. The application of integral equation methods to the numerical solution of boundary value problems. *Proc. Roy. Soc., London*, A232:201–210, 1971.
56. P. Businger and G. H. Golub. Handbook series linear algebra. Linear least squares solutions by Householder transformations. *Numer. Math.*, 7:269–276, 1965.
57. D. R. Butenhof. *Programming with POSIX threads*. Addison-Wesley, 1997.
58. F. Cakoni, G. C. Hsiao, and W. L. Wendland. On the boundary integral equation method for a mixed boundary value problem of the biharmonic equation. *Complex Var. Theory Appl.*, 50(7-11):681–696, 2005.
59. E. Van Camp, N. Mastronardi, and M. Van Barel. Two fast algorithms for solving diagonal-plus-semiseparable linear systems. *J. Comput. Appl. Math.*, 164–165:731–747, 2004.
60. S. Chandrasekaran and M. Gu. Fast and stable algorithms for banded plus semiseparable systems of linear equations. *SIAM J. Matrix Anal. Appl.*, 25:373–384, 2003.
61. S. Chandrasekaran, M. Gu, and W. Lyons. A fast adaptive solver for hierarchically semiseparable representations. *Calcolo*, 42(3-4):171–185, 2005.
62. S. Chandrasekaran, M. Gu, and T. Pals. Fast and stable algorithms for hierarchically semiseparable representations. Technical report, 2003.
63. F. W. Chapman. *Generalized orthogonal series for natural tensor product interpolation*. PhD thesis, University of Waterloo, 2003.
64. P. Chatzipantelidis, V. Ginting, and R. D. Lazarov. A finite volume element method for a nonlinear elliptic problem. *Numer. Linear Algebra Appl.*, 12(5-6):515–546, 2005.
65. G. Chen and J. Zhou. *Boundary Element Methods*. Academic Press, 1992.
66. Y. Chen. A fast, direct algorithm for the Lippmann-Schwinger integral equation in two dimensions. *Adv. Comput. Math.*, 16(2-3):175–190, 2002. Modeling and computation in optics and electromagnetics.
67. E. Chow. A priori sparsity patterns for parallel sparse approximate inverse preconditioners. *SIAM J. Sci. Comput.*, 21:1804–1822, 2000.
68. E. Chow and Y. Saad. Approximate inverse preconditioners via sparse-sparse iterations. *SIAM J. Sci. Comput.*, 19(3):995–1023 (electronic), 1998.
69. P. G. Ciarlet. *The finite element method for elliptic problems*. North-Holland Publishing Co., Amsterdam, 1978. Studies in Mathematics and its Applications, Vol. 4.

70. M. Costabel. Symmetric methods for the coupling of finite elements and boundary elements. In C. A. Brebbia, G. Kuhn, and W. L. Wendland, editors, *Boundary Elements IX*, pages 411–420. Springer-Verlag, Berlin, 1987.
71. M. Costabel and W. L. Wendland. Strong ellipticity of boundary integral operators. *J. Reine Angew. Math.*, 372:34–63, 1986.
72. E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 24th National Conference of the Association for Computing Machinery*, pages 157–172. Brondon Press, August 26–28 1969.
73. W. Dahmen, S. Prössdorf, and R. Schneider. Wavelet approximation methods for pseudodifferential equations. II. Matrix compression and fast solution. *Adv. Comput. Math.*, 1(3-4): 259–335, 1993.
74. E. Darve. The fast multipole method: numerical implementation. *J. Comput. Phys.*, 160(1):195–240, 2000.
75. T. A. Davis. *Direct methods for sparse linear systems*, volume 2 of *Fundamentals of Algorithms*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2006.
76. R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM J. Numer. Anal.*, 19:400–408, 1982.
77. S. Demko, W. F. Moss, and P. W. Smith. Decay rates for inverses of band matrices. *Math. Comp.*, 43(168):491–499, 1984.
78. J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Analysis and Applications*, 20:720–755, 1999.
79. J. W. Demmel, N. J. Higham, and R. Schreiber. Stability of block *LU* factorization. *Numer. Linear Algebra Appl.*, 2:173–190, 1995.
80. J. E. Dennis, Jr. and J. J. Moré. Quasi-Newton methods, motivation and theory. *SIAM Rev.*, 19(1):46–89, 1977.
81. J. E. Dennis, Jr. and Robert B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*, volume 16 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996. Corrected reprint of the 1983 original.
82. P. Deuflhard. *Newton Methods for Nonlinear Problems. Affine Invariance and Adaptive Algorithms*, volume 35 of *Series Computational Mathematics*. Springer, 2004.
83. P. Deuflhard, R. W. Freund, and A. Walter. Fast secant methods for the iterative solution of large nonsymmetric linear systems. *Impact Comput. Sci. Engrg.*, 2:244–276, 1990.
84. G. Dolzmann and S. Müller. Estimates for Green's matrices of elliptic systems by L^p theory. *Manuscripta Math.*, 88:261–273, 1995.
85. G. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrica*, 1:211–218, 1936.
86. Y. Eidelmann and I. Gohberg. Inversion formulas and linear complexity algorithm for diagonal plus semiseparable matrices. *Comp. Math. Appl.*, 33:69–79, 1997.
87. Y. Eidelmann and I. Gohberg. A look-ahead block schur algorithm for diagonal plus semiseparable matrices. *Comp. Math. Appl.*, 35:25–34, 1998.
88. M. A. Epton and B. Dembart. Multipole translation theory for the three-dimensional Laplace and Helmholtz equations. *SIAM J. Sci. Comput.*, 16(4):865–897, 1995.
89. C. Farhat and F.-X. Roux. A method of finite element tearing and interconnecting and its parallel solution algorithm. *Int. J. Numer. Meth. Engrg.*, 32:1205–1227, 1991.
90. T. Feder and D. Greene. Optimal algorithms for approximate clustering. In *Proc. 20th ACM Symp. Theory of Computing*, pages 434–444, 1988.
91. M. Fiedler. Algebraic connectivity of graphs. *Czech. Math. J.*, 23:298–305, 1973.
92. M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czech. Math. J.*, 25:619–637, 1975.
93. S. Funken and E. P. Stephan. The BPX preconditioner for the single layer potential operator. *Appl. Anal.*, 67(3–4):327–340, 1997.
94. D. Gaier. *Vorlesungen über Approximation im Complexen*. Birkhäuser Verlag, 1980.

95. F. R. Gantmacher and M. G. Krein. *Oszillationsmatrizen, Oszillationskerne und kleine Schwingungen mechanischer Systeme*. Akademie Verlag, Berlin, 1960. (in Russian); first edition, 1941.
96. I. Gavrilyuk, W. Hackbusch, and B. Khoromskij. Data-sparse approximation to operator-valued functions of elliptic operators. *Math. Comp.*, 73:1297–1324, 2004.
97. I. Gavrilyuk, W. Hackbusch, and B. Khoromskij. Data-sparse approximation of a class of operator-valued functions. *Math. Comp.*, 74:681–708, 2005.
98. I. Gavrilyuk, W. Hackbusch, and B. N. Khoromskij. $\mathscr{H}$-matrix approximation for the operator exponential with applications. *Numerische Mathematik*, 92:83–111, 2002.
99. A. George. Nested dissection of a regular finite element mesh. *SIAM J. Numer. Anal.*, 10:345–363, 1973.
100. A. George and J. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1981.
101. M. Giaquinta. *Multiple Integrals in the calculus of variations and nonlinear elliptic systems*. Annals of Mathematics Studies. Princeton University press, 1983.
102. K. Giebermann. *Schnelle Summationsverfahren zur numerischen Lösung von Integralgleichungen für Streuprobleme im* $\mathbb{R}^3$. PhD thesis, Universität Karlsruhe, 1997.
103. K. Giebermann. Multilevel approximation of boundary integral operators. *Computing*, 67:183–207, 2001.
104. D. Gilbarg and N. S. Trudinger. *Elliptic partial differential equations of second order*. Springer, Berlin, 2001. Reprint of the 1998 edition.
105. E. De Giorgi. Un esempio di estremali discontinue per un problema variazionale di tipo ellittico. *Boll. UMI*, 4:135–137, 1968.
106. G. H. Golub and Ch. F. Van Loan. *Matrix computations*. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.
107. T. Gonzales. Clustering to minimize the maximum intracluster distance. *Theoretical Computer Science*, 38:293–306, 1985.
108. D. Gope and V. Jandhyala. Oct-tree-based multilevel low-rank decomposition algorithm for rapid 3-d parasitic extraction. 23:1575–1580, 2004.
109. S. A. Goreinov. Mosaic-skeleton approximation of matrices generated by asymptotically smooth and oscillatory kernels. In E. E. Tyrtyshnikov, editor, *Matrix Methods and Algorithms*, pages 42–76. INM RAS, Moscow, 1999. (in Russian).
110. S. A. Goreinov, E. E. Tyrtyshnikov, and A. Yu. Yeremin. Matrix-free iterative solution strategies for large dense linear systems. *Numer. Linear Algebra Appl.*, 4(4):273–294, 1997.
111. S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin. A theory of pseudoskeleton approximations. *Linear Algebra Appl.*, 261:1–21, 1997.
112. I. G. Graham and M. J. Hagger. Unstructured additive Schwarz-conjugate gradient method for elliptic problems with highly discontinuous coefficients. *SIAM J. Sci. Comput.*, 20(6):2041–2066 (electronic), 1999.
113. R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal of Applied Mathematics*, 17(2):416–429, 1969.
114. L. Grasedyck. *Theorie und Anwendungen Hierarchischer Matrizen*. PhD thesis, Universität Kiel, 2001.
115. L. Grasedyck. Adaptive recompression of $\mathscr{H}$-matrices for BEM. *Computing*, 74:205–223, 2005.
116. L. Grasedyck and W. Hackbusch. Construction and arithmetics of $\mathscr{H}$-matrices. *Computing*, 70:295–334, 2003.
117. A. Greenbaum. *Iterative methods for solving linear systems*. Number 17 in Frontiers in Applied Mathematics. SIAM, Philadelphia, PA, 1997.
118. L. Greengard. *The rapid evaluation of potential fields in particle systems*. MIT Press, Cambridge, MA, 1988.
119. L. Greengard and J. Strain. The fast Gauss transform. *SIAM J. Sci. Statist. Comput.*, 12(1): 79–94, 1991.

120. L. Greengard and X. Sun. A new version of the fast Gauss transform. In *Proceedings of the International Congress of Mathematicians, Vol. III (Berlin, 1998)*, number Extra Vol. III, pages 575–584 (electronic), 1998.
121. L. F. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73(2):325–348, 1987.
122. L. F. Greengard and V. Rokhlin. A new version of the fast multipole method for the Laplace equation in three dimensions. In *Acta numerica, 1997*, volume 6 of *Acta Numer.*, pages 229–269. Cambridge Univ. Press, Cambridge, 1997.
123. M. J. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM J. Sci. Comput.*, 18:838–853, 1997.
124. M. Grüter and K.-O. Widman. The Green function for uniformly elliptic equations. *Manuscripta Math.*, 37(3):303–342, 1982.
125. W. Hackbusch. *Multi-Grid Methods and Applications*. Springer, 1985.
126. W. Hackbusch. *Theorie und Numerik elliptischer Differentialgleichungen*. B. G. Teubner, Stuttgart, second edition, 1996.
127. W. Hackbusch. A sparse matrix arithmetic based on $\mathscr{H}$-matrices. Part I: Introduction to $\mathscr{H}$-matrices. *Computing*, 62(2):89–108, 1999.
128. W. Hackbusch. Hierarchische Matrizen – Algorithmen und Analysis. Lecture Notes, September 2004.
129. W. Hackbusch and S. Börm. Data-sparse approximation by adaptive $\mathscr{H}^2$-matrices. *Computing*, 69(1):1–35, 2002.
130. W. Hackbusch and S. Börm. $\mathscr{H}^2$-matrix approximation of integral operators by interpolation. *Appl. Numer. Math.*, 43(1-2):129–143, 2002. 19th Dundee Biennial Conference on Numerical Analysis (2001).
131. W. Hackbusch and B. N. Khoromskij. $\mathscr{H}$-matrix approximation on graded meshes. In John R. Whiteman, editor, *The Mathematics of Finite Elements and Applications X*, pages 307–316. Elsevier, 2000.
132. W. Hackbusch and B. N. Khoromskij. A sparse $\mathscr{H}$-matrix arithmetic: general complexity estimates. *J. Comput. Appl. Math.*, 125(1-2):479–501, 2000. Numerical analysis 2000, Vol. VI, Ordinary differential equations and integral equations.
133. W. Hackbusch and B. N. Khoromskij. A sparse $\mathscr{H}$-matrix arithmetic. Part II: Application to multi-dimensional problems. *Computing*, 64(1):21–47, 2000.
134. W. Hackbusch, B. N. Khoromskij, and R. Kriemann. Hierarchical matrices based on a weak admissibility criterion. *Computing*, 73:207–243, 2004.
135. W. Hackbusch, B. N. Khoromskij, and R. Kriemann. Direct Schur complement method by domain decomposition based on $\mathscr{H}$-matrix approximation. *Comput. Vis. Sci.*, 8(3-4): 179–188, 2005.
136. W. Hackbusch, B. N. Khoromskij, and S. A. Sauter. On $\mathscr{H}^2$-matrices. In H.-J. Bungartz, R. H. W. Hoppe, and Ch. Zenger, editors, *Lectures on Applied Mathematics*, pages 9–29. Springer-Verlag, Berlin, 2000.
137. W. Hackbusch and Z. P. Nowak. On the complexity of the panel method. In *International Conference on Modern Problems in Numerical Analysis*. Moscow, 1986.
138. W. Hackbusch and Z. P. Nowak. On the fast matrix multiplication in the boundary element method by panel clustering. *Numer. Math.*, 54(4):463–491, 1989.
139. W. Hackbusch and S. A. Sauter. On the efficient use of the Galerkin method to solve Fredholm integral equations. In *Proceedings of ISNA '92—International Symposium on Numerical Analysis, Part I (Prague, 1992)*, volume 38, pages 301–322, 1993.
140. S. Hamada and A. Tatematsu. Solving regularized least squares with qualitatively controlled adaptive cross-approximated matrices. *Electrical Engineering in Japan*, 159(3), 2007. translated from Denki Gakkai Ronbunshi, Vol. 125-A, No. 5, May 2005, pp. 419–426.
141. H. Han. The boundary integro-differential equations of three-dimensional neumann problem in linear elasticity. *Numer. Math.*, 68:269–281, 1994.
142. H. Harbrecht, U. Kähler, and R. Schneider. Wavelet Galerkin BEM on unstructured meshes. *Comput. Vis. Sci.*, 8:189–199, 2005.

143. K. Hayami and S. A. Sauter. A panel clustering method for 3-D elastostatics using spherical harmonics. In *Integral methods in science and engineering (Houghton, MI, 1998)*, pages 179–184. Chapman & Hall/CRC, Boca Raton, FL, 2000.
144. B. Hendrickson and E. Rothberg. Improving the run time and quality of nested disection ordering. *SIAM J. Sci. Comp.*, 20:468–489, 1998.
145. M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Research Nat. Bur. Standards*, 49:409–436 (1953), 1952.
146. R. Hiptmair. Finite elements in computational electromagnetism. *Acta Numerica*, pages 237–339, 2002.
147. J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.
148. R. A. Horn and Ch. R. Johnson. *Matrix analysis*. Cambridge University Press, Cambridge, 1990. Corrected reprint of the 1985 original.
149. R. A. Horn and Ch. R. Johnson. *Topics in matrix analysis*. Cambridge University Press, Cambridge, 1994. Corrected reprint of the 1991 original.
150. Jr. J. Douglas and T. Dupont. A Galerkin method for a nonlinear Dirichlet problem. *Math. Comp.*, 29:689–696, 1975.
151. P. Jones, J. Ma, and V. Rokhlin. A Fast Direct Algorithm for the Solution of the Laplace Equation on Regions with Fractal Boundaries. *J. Comp. Phys.*, 113:35–51, 1994.
152. S. Kapur and D. E. Long. IES^3: A fast integral equation solver for efficient 3-dimensional extraction. In *Proc. ICCAD*, pages 448–455, 1997.
153. J. Karátson and I. Faragó. Variable preconditioning via quasi-Newton methods for nonlinear problems in Hilbert space. *SIAM J. Numer. Anal.*, 41:1242–1262, 2003.
154. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1999.
155. C. T. Kelley. *Solving nonlinear equations with Newton's method*. Fundamentals of Algorithms. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2003.
156. B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 29:291–307, 1970.
157. B. N. Khoromskij and J. M. Melenk. Boundary concentrated finite element methods. *SIAM J. Numer. Anal.*, 41(1):1–36 (electronic), 2003.
158. B. N. Khoromskij and W. L. Wendland. Spectral equivalent preconditioners for boundary equations in substructuring techniques. *East-West Journal of Numer. Math.*, 1(1):1–26, 1992.
159. B. N. Khoromskij and G. Wittum. Robust Schur complement method for strongly anisotropic elliptic equations. *Numer. Linear Algebra Appl.*, 6:1–33, 1999.
160. B. N. Khoromskij and G. Wittum. *Numerical solution of elliptic differential equations by reduction to the interface*, volume 36 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Berlin, 2004.
161. Boris N. Khoromskij and Siegfried Prössdorf. Fast computations with the harmonic Poincaré-Steklov operators on nested refined meshes. *Adv. Comput. Math.*, 8(1-2):111–135, 1998.
162. L. Yu. Kolotilina, A. A. Nikishin, and A. Yu. Yeremin. Factorized sparse approximate inverse preconditionings. IV. Simple approaches to rising efficiency. *Numer. Linear Algebra Appl.*, 6:515–531, 1999.
163. L. Yu. Kolotilina and A. Yu. Yeremin. Factorized sparse approximate inverse preconditionings. I. Theory. *SIAM J. Matrix Anal. Appl.*, 14:45–58, 1993.
164. H. König. An explicit formula for fundamental solutions of linear partial differential equations with constant coefficients. *Proc. Amer. Math. Soc.*, 120(4):1315–1318, 1994.
165. R. Kriemann. *Implementation and Usage of a Thread Pool based on POSIX Threads*. MPI MiS Leipzig, 2003. Report 2/2003.
166. R. Kriemann. Parallel $\mathscr{H}$-matrix arithmetics on shared memory systems. *Computing*, 74(3):273–297, 2005.
167. V. Kupradze. *Three–dimensional problems of the mathematical theory of elasticity and thermoelasticity*. North–Holland, Amsterdam, 1979.

168. S. Kurz, O. Rain, and S. Rjasanow. The adaptive cross approximation technique for the 3d boundary element method. *IEEE Transaction on Magnetics*, 38(2):421–424, 2002.
169. S. Kurz, O. Rain, and S. Rjasanow. Application of the adaptive cross approximation technique for the coupled BE-FE solution of symmetric electromagnetic problems. *Computational Mechanics*, 32:423–429, 2003.
170. P. K. Kythe. *Fundamental solutions for Differential Operators and Applications*. Birkhäuser, Boston, 1996.
171. Ch. Lage and Ch. Schwab. Wavelet Galerkin algorithms for boundary integral equations. *SIAM J. Sci. Comput.*, 20(6):2195–2222 (electronic), 1999.
172. U. Langer and D. Pusch. Data-sparse algebraic multigrid methods for large scale boundary element equations. *Appl. Numer. Math.*, 54(3-4):406–424, 2005.
173. U. Langer, D. Pusch, and S. Reitzinger. Efficient preconditioners for boundary element matrices based on grey-box algebraic multigrid methods. *Int. J. Numer. Meth. Engng.*, 58: 1937–1953, 2003.
174. U. Langer and O. Steinbach. Boundary Element Tearing and Interconnecting Methods. *Computing*, 71:205–228, 2003.
175. S. Le Borne. $\mathscr{H}$-matrices for convection-diffusion problems with constant convection. *Computing*, 70:261–274, 2003.
176. M. Lintner. *Lösung der 2D Wellengleichung mittels hierarchischer Matrizen*. PhD thesis, Technische Universität München, Germany, 2002.
177. M. Maischak, E. P. Stephan, and T. Tran. Multiplicative Schwarz algorithms for the Galerkin boundary element method. *SIAM J. Numer. Anal.*, 38(4):1243–1268, 2000.
178. P. G. Martinsson and V. Rokhlin. A fast direct solver for boundary integral equations in two dimensions. *J. Comp. Phys.*, 205:1–23, 2005.
179. N. Mastronardi, S. Chandrasekaran, and S. Van Huffel. Fast and stable two-way algorithm for diagonal plus semi-separable systems of linear equations. *Num. Lin. Alg. Appl.*, 8:7–12, 2001.
180. A. W. Maue. Zur Formulierung eines allgemeinen Beugungsproblems durch eine Integralgleichung. *Z. f. Physik*, 126:601–618, 1949.
181. W. F. McColl. Scalable computing. In *Computer Science Today: Recent Trends and Developments*, volume 1000 of *LNCS*, pages 46–61. Springer-Verlag, 1995.
182. W. McLean. *Strongly Elliptic Systems and Boundary Integral Equations*. Cambridge University Press, 2000.
183. G. Meinardus. *Approximation of functions: Theory and numerical methods*. Springer-Verlag, New York, 1967.
184. J. M. Melenk, S. Börm, and M. Löhndorf. Approximation of integral operators by variable-order interpolation. *Numerische Mathematik*, 99:605–643, 2005.
185. A. Meyer and S. Rjasanow. An effective direct solution method for certain boundary element equations in 3D. *Math. Methods Appl. Sci.*, 13(1):43–53, 1990.
186. E. Michielssen and A. Boag. A multilevel matrix decomposition for analyzing scattering from large structures. *IEEE Trans. Antennas Propag.*, 44:1086–1093, 1996.
187. G. L. Miller, S.-H. Teng, W. Thurston, and S. A. Vavasis. Geometric separators for finite-element meshes. *SIAM J. Sci. Comput.*, 19, 1998.
188. L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *Quart. J. Math. Oxford Ser. (2)*, 11:50–59, 1960.
189. R. Nabben. Decay rates of the inverse of nonsymmetric tridiagonal and banded matrices. *SIAM J. Matrix Anal. Appl.*, 20:820–837, 1999.
190. J. Nédélec. Integral equations with non-integrable kernels. *Integral Equ. Oper. Theory*, 5:562–572, 1982.
191. N. Neuss, W. Jäger, and G. Wittum. Homogenization and multigrid. *Computing*, 66:1–26, 2001.
192. N. Nishimura. Fast multipole accelerated boundary integral equation methods. *Appl. Mech. Rev.*, 55(4):299–324, 2002.

193. G. Of, O. Steinbach, and W. Wendland. The fast multipole method for the symmetric boundary integral formulation. *IMA Journal of Numerical Analysis*, 26:272–296, 2006.
194. G. Of, O. Steinbach, and W. L. Wendland. Applications of a fast multipole Galerkin boundary element method in linear elastostatics. *Comput. Vis. Sci.*, 8(3-4):201–209, 2005.
195. B. Olstad and F. Manne. Efficient partitioning of sequences. *IEEE Trans. Comput.*, 44: 1322–1326, 1995.
196. J. Ostrowski, Z. Andjelić, M. Bebendorf, B. Crânganu-Creţu, and J. Smajić. Fast BEM-Solution of Laplace Problems with $\mathscr{H}$-Matrices and ACA. *IEEE Trans. on Magnetics*, 42(4):627–630, 2006.
197. J. Ostrowski, R. Hiptmair, and H. Fuhrmann. Electric 3d-simulation of metallized film capacitors. *COMPEL*, 26:524–543, 2007.
198. N. A. Ozdemir and J. F. Lee. A low rank IE-*QR* algorithm for matrix compression in volume integral equations. *IEEE Trans. Magn.*, 40:1017–1020, 2004.
199. C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Num. Anal.*, 12:617–629, 1975.
200. L. E. Payne and H. F. Weinberger. An optimal Poincaré inequality for convex domains. *Arch. Rational Mech. Anal.*, 5:286–292, 1960.
201. K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.
202. M. Pester and S. Rjasanow. A parallel preconditioned iterative realization of the panel method in 3D. *Numer. Linear Algebra Appl.*, 3(1):65–80, 1996.
203. A. Huard Ph. Guillaume and C. Le Calvez. A block constant approximate inverse for preconditioning large linear systems. *SIAM J. Matrix Anal. Appl.*, 24:822–851, 2003.
204. J. R. Phillips and J. White. A precorrected-FFT method for electrostatic analysis of complicated 3-d structures. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 16:1059–1072, 1997.
205. A. Pothen, H. D. Simon, and L. Wang. Spectral Nested Dissection. Technical Report RNR-92-003, Nasa Ames Research Center, Moffett Field, CA 94035, 1992.
206. S. Prössdorf and B. Silbermann. *Numerical analysis for integral and related operator equations*. Akademie Verlag, Berlin, 1991.
207. J. Rahola. Diagonal forms of the translation operators in the fast multipole algorithm for scattering problems. *BIT*, 36(2):333–358, 1996.
208. V. C. Raykar, C. Yang, R. Duraiswami, and N. Gumerov. Fast computation of sums of Gaussians in high dimensions. Technical report, Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, 2005.
209. J. M. Rius, J. Parrón, E. Úbeda, and J. R. Mosig. Multilevel matrix decomposition for analysis of electrically large electromagnetic problems in 3-d. *Microwave Opt. Tech. Lett.*, 22: 177–182, 1999.
210. S. Rjasanow. Effective algorithms with circulant-block matrices. *Linear Algebra Appl.*, 202:55–69, 1994.
211. S. Rjasanow. Optimal preconditioner for boundary element formulation of the Dirichlet problem in elasticity. *Math. Methods Appl. Sci.*, 18(8):603–613, 1995.
212. S. Rjasanow. The structure of the boundary element matrix for the three-dimensional Dirichlet problem in elasticity. *Numer. Linear Algebra Appl.*, 5(3):203–217, 1998.
213. S. Rjasanow, I. Ibraghimov, and K. Straube. Hierarchical Cholesky decomposition of sparse matrices arising from curl-curl-equations. *J. Numer. Math.*, 15(1):31–58, 2007.
214. S. Rjasanow and O. Steinbach. *The fast solution of boundary integral equations. Mathematical and analytical techniques with appications in engineering*. Springer, New York, 2007.
215. V. Rokhlin. Rapid solution of integral equations of classical potential theory. *J. Comput. Phys.*, 60(2):187–207, 1985.
216. V. Rokhlin. Rapid solution of integral equations of scattering theory in two dimensions. *J. Comput. Phys.*, 86(2):414–439, 1990.
217. V. Rokhlin. Diagonal forms of translation operators for the Helmholtz equation in three dimensions. *Appl. Comput. Harmon. Anal.*, 1(1):82–93, 1993.

218. D. J. Rose. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. In R. C. Read, editor, *Graph Theory and Computing*, pages 183–217. Academic Press, 1972.
219. P. Rósza. On the inverse of band matrices. *Integral Equations and Operator theory*, 10: 82–95, 1987.
220. J. W. Ruge and K. Stüben. Algebraic multigrid. In S. F. McCormick, editor, *Multigrid Methods*, pages 73–130. SIAM, 1987.
221. Y. Saad. Krylov subspace methods: theory, algorithms, and applications. In *Computing methods in applied sciences and engineering (Paris, 1990)*, pages 24–41. SIAM, Philadelphia, PA, 1990.
222. Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing, Boston, 1996.
223. Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 2003.
224. Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7(3):856–869, 1986.
225. Th. Sauer and Y. Xu. On multivariate Lagrange interpolation. *Math. Comp.*, 64(211): 1147–1170, 1995.
226. S. A. Sauter. *Über die Verwendung des Galerkinverfahrens zur Lösung Fredholmscher Integralgleichungen*. PhD thesis, Universität Kiel, 1992.
227. S. A. Sauter. Variable order panel clustering. *Computing*, 64:223–261, 2000.
228. S. A. Sauter and Ch. Schwab. *Randelementmethoden*. Teubner, 2004.
229. A. H. Schatz, V. Thomée, and W. L. Wendland. *Mathematical theory of finite and boundary element methods*. Birkhäuser Verlag, Basel, 1990.
230. O. Schenk and K. Gärtner. Solving unsymmetric sparse systems of linear equations with PARDISO. In *Computational science—ICCS 2002, Part II (Amsterdam)*, volume 2330 of *Lecture Notes in Comput. Sci.*, pages 355–363. Springer, Berlin, 2002.
231. E. Schmidt. Zur Theorie der linearen und nichtlinearen Integralgleichungen. Teil 1. *Math. Annalen*, 63:433–476, 1907.
232. R. Schneider. *Multiskalen- und Wavelet-Matrixkompression*. B. G. Teubner, Stuttgart, 1998. Analysisbasierte Methoden zur effizienten Lösung großer vollbesetzter Gleichungssysteme.
233. J. Schöberl. NETGEN – an advancing front 2d/3d-mesh generator based on abstract rules. *Comput. Visual. Sci*, 1:41–52, 1997.
234. A. Schönhage. *Approximationstheorie*. de Gruyter, Berlin, 1971.
235. G. Schulz. Iterative Berechnung der reziproken Matrix. *ZAMM*, 13, 1933.
236. S. Sirtori. General stress analysis method by means of integral equations and boundary elements. *Meccanica*, 14:210–218, 1979.
237. S. Sirtori, G. Maier, G. Novati, and S. Miccoli. A Galerkin symmetric boundary-element method in elasticity: formulation and implementation. *Internat. J. Numer. Methods Engrg.*, 35:255–282, 1992.
238. B. F. Smith, P. E. Bjørstad, and W. D. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge Univ. Press, 1996.
239. P. Starr and V. Rokhlin. On the Numerical Solution of Two-Point Boundary Value Problems II. *Comm. Pure and Appl. Math.*, 47:1117–1159, 1994.
240. O. Steinbach. *Gebietszerlegungsmethoden mit Randintegralgleichungen und effiziente numerische Lösungsverfahren für gemischte Randwertprobleme*. PhD thesis, Universität Stuttgart, Germany, 1996.
241. O. Steinbach. Artificial multilevel boundary element preconditioners. *Proc. Appl. Math. Mech.*, 3:539–542, 2003.
242. O. Steinbach. *Numerische Näherungsverfahren für elliptische Randwertprobleme: Finite Elemente und Randelemente*. B. G. Teubner, Stuttgart, Leipzig, Wiesbaden, 2003.
243. O. Steinbach. *Stability estimates for hybrid coupled domain decomposition methods*, volume 1809 of *Springer Lecture Notes in Mathematics*. Springer, Berlin, 2003.
244. O. Steinbach and W. Wendland. The construction of some efficient preconditioners in the boundary element method. *Adv. Comput. Math.*, 9:191–216, 1998.

245. T. Steinmetz, N. Gödel, G. Wimmer, M. Clemens, S. Kurz, and M. Bebendorf. Efficient symmetric FEM-BEM coupled simulations of electro-quasistatic fields. *IEEE Trans. Magn.*, 2007. to appear.
246. T. Steinmetz, N. Gödel, G. Wimmer, M. Clemens, S. Kurz, M. Bebendorf, and S. Rjasanow. Symmetric coupling of the finite-element and the boundary-element method for electro-quasistatic field simulations. In G. Ciuprina and D. Ioan, editors, *Scientific Computing in Electrical Engineering*, pages 309–315. Springer-Verlag, Berlin, 2007.
247. F. Stenger. *Numerical methods based on Sinc and analytic functions*. Springer Verlag, Heidelberg, 1993.
248. M. Stolper. *Schnelle Randelementmethoden für die Helmholtz-Gleichung*. PhD thesis, Saarland University, 2004.
249. R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.
250. J. Tausch. The variable order fast multipole method for boundary integral equations of the second kind. *Computing*, 72:267–291, 2004.
251. J. Tausch and J. White. Multiscale bases for the sparse representation of boundary integral operators on complex geometries. *SIAM J. Sci. Comput.*, 24:1610–1629, 2003.
252. E. E. Tyrtyshnikov. Mosaic-skeleton approximations. *Calcolo*, 33(1-2):47–57 (1998), 1996. Toeplitz matrices: structures, algorithms and applications (Cortona, 1996).
253. L. G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990.
254. R. Vandebril, M. Van Barel, and N. Mastronardi. A note on the representation and definition of semiseparable matrices. *Numer. Linear Algebra Appl.*, 12(8):839–858, 2005.
255. T. von Petersdorff and Ch. Schwab. Wavelet approximations for first kind boundary integral equations on polygons. *Numer. Math.*, 74(4):479–516, 1996.
256. T. von Petersdorff, Ch. Schwab, and R. Schneider. Multiwavelets for second-kind integral equations. *SIAM J. Numer. Anal.*, 34(6):2212–2227, 1997.
257. T. von Petersdorff and E. P. Stephan. On the convergence of the multigrid method for a hypersingular integral equation of the first kind. *Numer. Math.*, 57:379–391, 1990.
258. T. von Petersdorff and E. P. Stephan. Multigrid solvers and preconditioners for first kind integral equations. *Numer. Methods Partial Differ. Equations*, 8(5):443–450, 1992.
259. W. L. Wendland. On some mathematical aspects of boundary element methods for elliptic problems. In *The mathematics of finite elements and applications, V (Uxbridge, 1984)*, pages 193–227. Academic Press, London, 1985.
260. W. L. Wendland. Bemerkungen zu Randelementmethoden und ihren mathematischen und numerischen Aspekten. *Mitteilungen der GAMM, Heft 2*, pages 3–27, 1986.
261. H. Yserentant. On the multi-level splitting of finite element spaces. *Numer. Math.*, 49: 379–412, 1986.
262. Ch. Zenger. Sparse grids. In W. Hackbusch, editor, *Parallel Algorithms for Partial Differential Equations*, volume 31 of *Notes on Numerical Fluid Mechanics*, pages 241–251. Vieweg, 1991.
263. K. Zhao, M. N. Vouvakis, and J.-F. Lee. The adaptive cross approximation algorithm for accelerated method of moments computation of EMC problems. *IEEE Transaction on Electromagnetic Compatibility*, 47:763–773, 2005.

Appendix

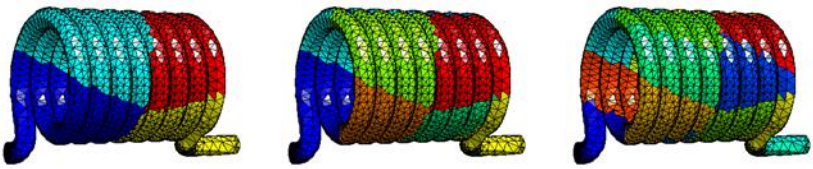


Fig. 1.4 Three levels in a cluster tree.

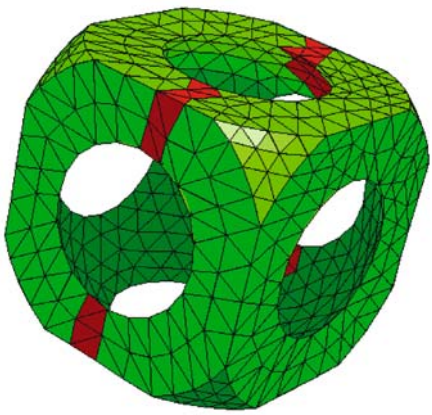


Fig. 1.8 Computational domain with an interface cluster.

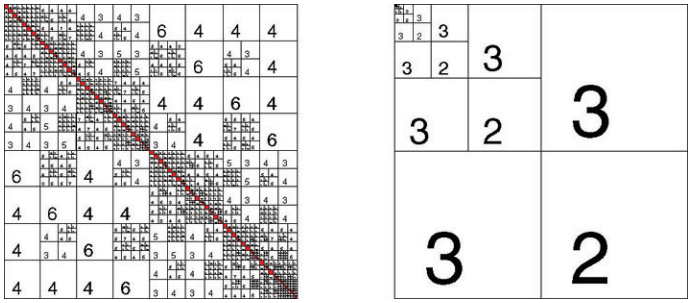


Fig. 2.1 $\mathscr{H}$-matrices with their blockwise ranks.

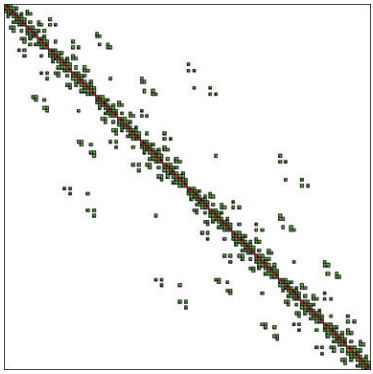


Fig. 2.2 A sparse $\mathscr{H}$-matrix (nonzero blocks are shown).

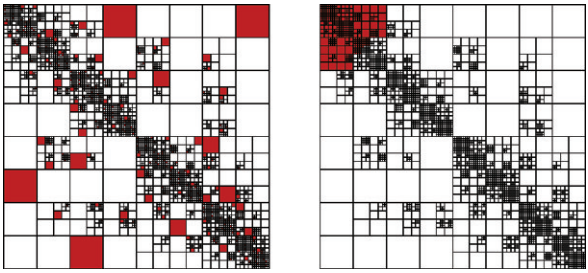


Fig. 2.4 List scheduling (left) versus sequence partitioning (right).

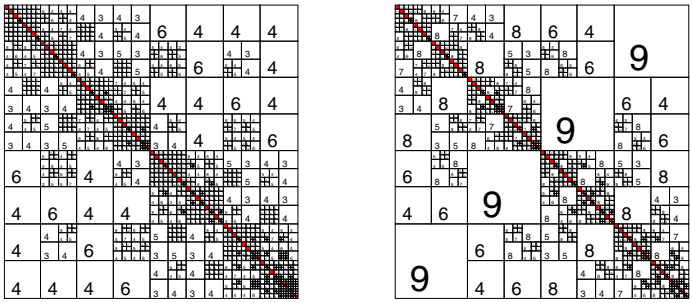


Fig. 2.9 $\mathscr{H}$-matrix before and after coarsening.

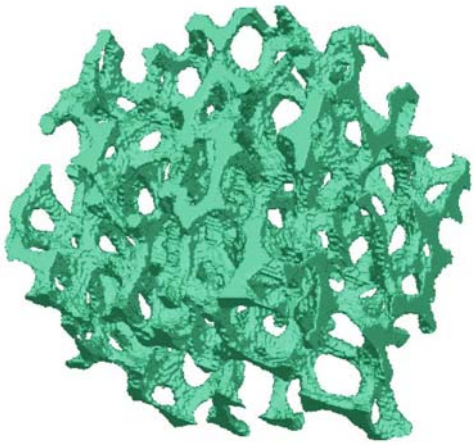


Fig. 3.4 Surface mesh of a foam with $n = 494\,616$ degrees of freedom.

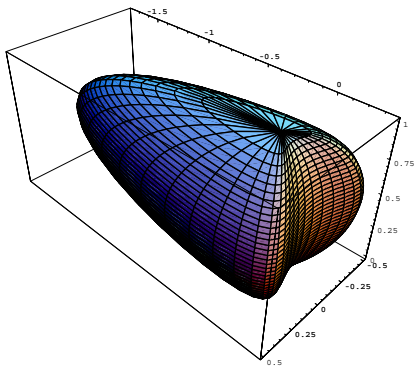


Fig. 3.7 The test surface.

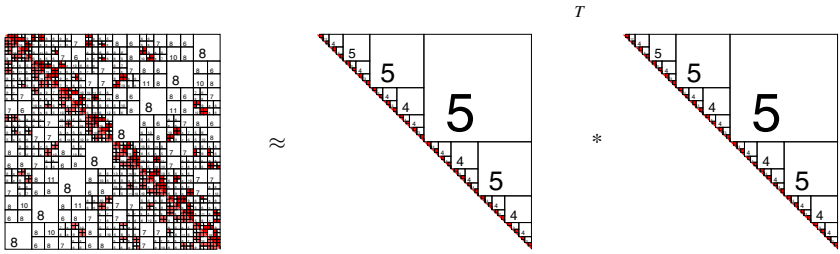


Fig. 3.8 Low-precision Cholesky decomposition.

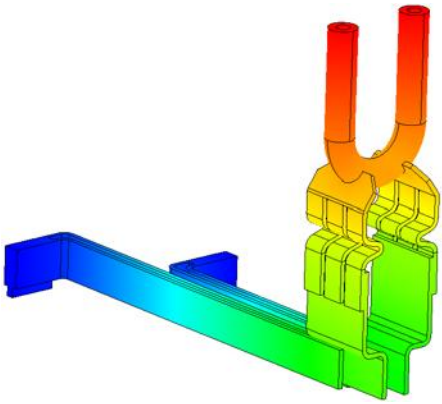


Fig. 3.9 Device with electric potential.

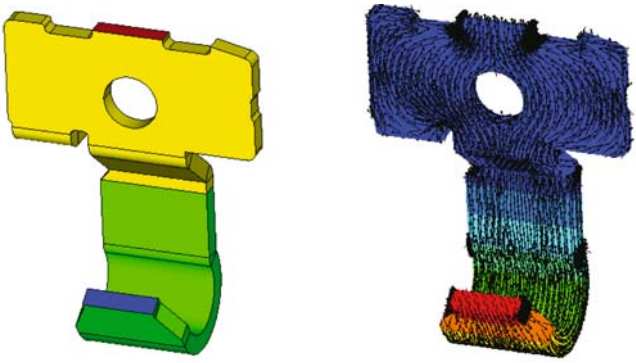


Fig. 3.10 Geometry with computed surface current.

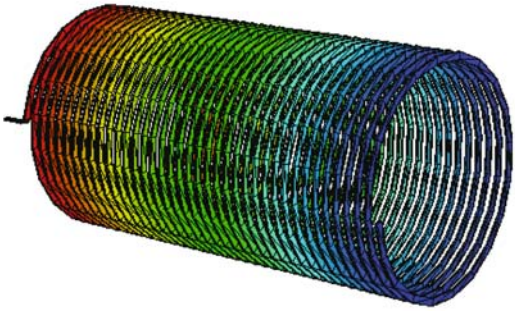


Fig. 3.11 Computational geometry with solution.

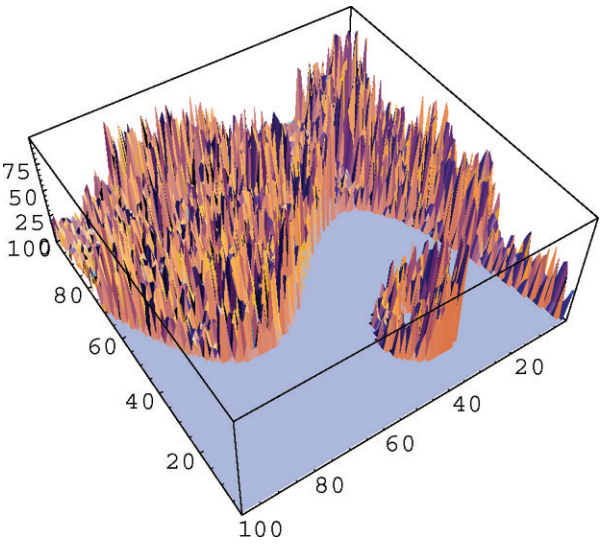


Fig. 4.2 The coefficient $\alpha(x)$.

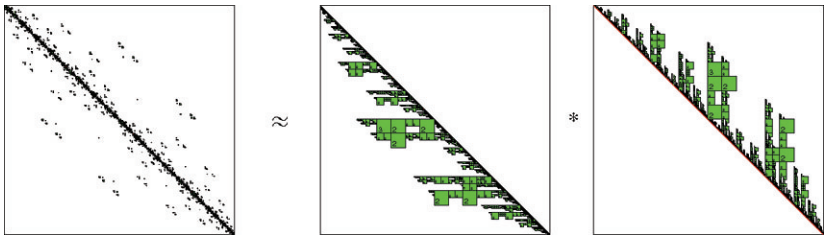


Fig. 4.7 Approximate LU decomposition.

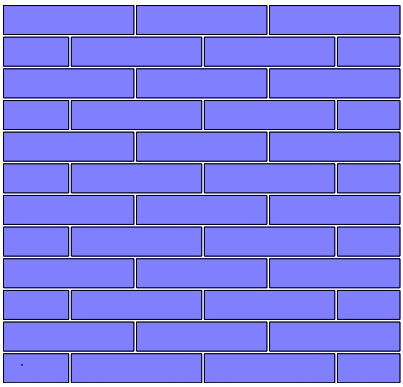


Fig. 4.9 Geometry of skin fragment with diffusion coefficient.

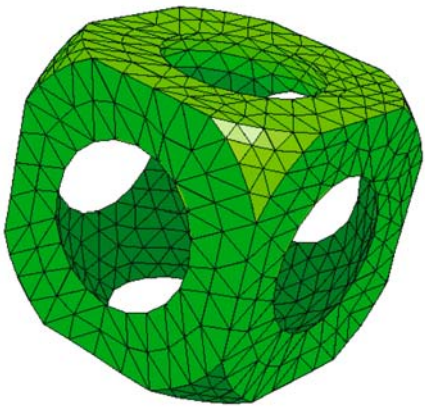


Fig. 4.10 The computational domain.

Fig. 4.11 Nested dissection Cholesky decomposition.

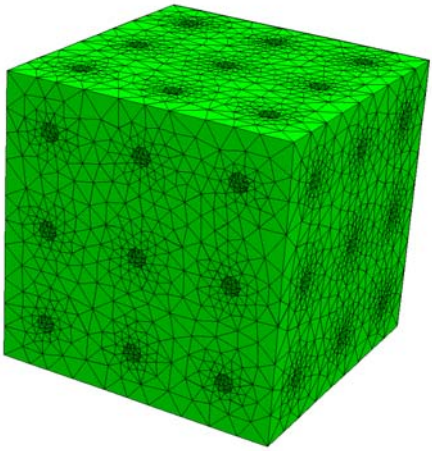


Fig. 4.12 Computational domain.

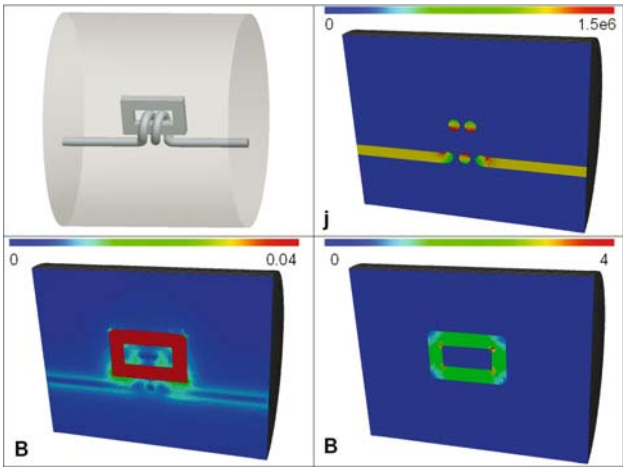


Fig. 4.15 Results of the magnetostatic field computations.

Index

Editorial Policy

1. Volumes in the following three categories will be published in LNCSE:

i) Research monographs
ii) Lecture and seminar notes
iii) Conference proceedings

Those considering a book which might be suitable for the series are strongly advised to contact the publisher or the series editors at an early stage.

2. Categories i) and ii). These categories will be emphasized by Lecture Notes in Computational Science and Engineering. **Submissions by interdisciplinary teams of authors are encouraged**. The goal is to report new developments – quickly, informally, and in a way that will make them accessible to non-specialists. In the evaluation of submissions timeliness of the work is an important criterion. Texts should be well-rounded, well-written and reasonably self-contained. In most cases the work will contain results of others as well as those of the author(s). In each case the author(s) should provide sufficient motivation, examples, and applications. In this respect, Ph.D. theses will usually be deemed unsuitable for the Lecture Notes series. Proposals for volumes in these categories should be submitted either to one of the series editors or to Springer-Verlag, Heidelberg, and will be refereed. A provisional judgment on the acceptability of a project can be based on partial information about the work: a detailed outline describing the contents of each chapter, the estimated length, a bibliography, and one or two sample chapters – or a first draft. A final decision whether to accept will rest on an evaluation of the completed work which should include

– at least 100 pages of text;
– a table of contents;
– an informative introduction perhaps with some historical remarks which should be accessible to readers unfamiliar with the topic treated;
– a subject index.

3. Category iii). Conference proceedings will be considered for publication provided that they are both of exceptional interest and devoted to a single topic. One (or more) expert participants will act as the scientific editor(s) of the volume. They select the papers which are suitable for inclusion and have them individually refereed as for a journal. Papers not closely related to the central topic are to be excluded. Organizers should contact Lecture Notes in Computational Science and Engineering at the planning stage.

In exceptional cases some other multi-author-volumes may be considered in this category.

4. Format. Only works in English are considered. They should be submitted in camera-ready form according to Springer-Verlag's specifications.
Electronic material can be included if appropriate. Please contact the publisher.
Technical instructions and/or LaTeX macros are available via http://www.springer.com/authors/book+authors?SGWID=0-154102-12-417900-0. The macros can also be sent on request.

General Remarks

Lecture Notes are printed by photo-offset from the master-copy delivered in camera-ready form by the authors. For this purpose Springer-Verlag provides technical instructions for the preparation of manuscripts. See also *Editorial Policy.*

Careful preparation of manuscripts will help keep production time short and ensure a satisfactory appearance of the finished book.

The following terms and conditions hold:

Categories i), ii), and iii):
Authors receive 50 free copies of their book. No royalty is paid. Commitment to publish is made by letter of intent rather than by signing a formal contract. Springer- Verlag secures the copyright for each volume.

For conference proceedings, editors receive a total of 50 free copies of their volume for distribution to the contributing authors.

All categories:
Authors are entitled to purchase further copies of their book and other Springer mathematics books for their personal use, at a discount of 33.3% directly from Springer-Verlag.

Addresses:

Timothy J. Barth
NASA Ames Research Center
NAS Division
Moffett Field, CA 94035, USA
e-mail: barth@nas.nasa.gov

Michael Griebel
Institut für Numerische Simulation
der Universität Bonn
Wegelerstr. 6
53115 Bonn, Germany
e-mail: griebel@ins.uni-bonn.de

David E. Keyes
Department of Applied Physics
and Applied Mathematics
Columbia University
200 S. W. Mudd Building
500 W. 120th Street
New York, NY 10027, USA
e-mail: david.keyes@columbia.edu

Risto M. Nieminen
Laboratory of Physics
Helsinki University of Technology
02150 Espoo, Finland
e-mail: rni@fyslab.hut.fi

Dirk Roose
Department of Computer Science
Katholieke Universiteit Leuven
Celestijnenlaan 200A
3001 Leuven-Heverlee, Belgium
e-mail: dirk.roose@cs.kuleuven.ac.be

Tamar Schlick
Department of Chemistry
Courant Institute of Mathematical
Sciences
New York University
and Howard Hughes Medical Institute
251 Mercer Street
New York, NY 10012, USA
e-mail: schlick@nyu.edu

Mathematics Editor at Springer:
Martin Peters
Springer-Verlag
Mathematics Editorial IV
Tiergartenstrasse 17
D-69121 Heidelberg, Germany
Tel.: *49 (6221) 487-8185
Fax: *49 (6221) 487-8355
e-mail: martin.peters@springer.com

Lecture Notes in Computational Science and Engineering

1. D. Funaro, *Spectral Elements for Transport-Dominated Equations.*
2. H. P. Langtangen, *Computational Partial Differential Equations.* Numerical Methods and Diffpack Programming.
3. W. Hackbusch, G. Wittum (eds.), *Multigrid Methods V.*
4. P. Deuflhard, J. Hermans, B. Leimkuhler, A. E. Mark, S. Reich, R. D. Skeel (eds.), *Computational Molecular Dynamics: Challenges, Methods, Ideas.*
5. D. Kröner, M. Ohlberger, C. Rohde (eds.), *An Introduction to Recent Developments in Theory and Numerics for Conservation Laws.*
6. S. Turek, *Efficient Solvers for Incompressible Flow Problems.* An Algorithmic and Computational Approach.
7. R. von Schwerin, ***M**ulti **B**ody **S**ystem **SIM**ulation.* Numerical Methods, Algorithms, and Software.
8. H.-J. Bungartz, F. Durst, C. Zenger (eds.), *High Performance Scientific and Engineering Computing.*
9. T. J. Barth, H. Deconinck (eds.), *High-Order Methods for Computational Physics.*
10. H. P. Langtangen, A. M. Bruaset, E. Quak (eds.), *Advances in Software Tools for Scientific Computing.*
11. B. Cockburn, G. E. Karniadakis, C.-W. Shu (eds.), *Discontinuous Galerkin Methods.* Theory, Computation and Applications.
12. U. van Rienen, *Numerical Methods in Computational Electrodynamics.* Linear Systems in Practical Applications.
13. B. Engquist, L. Johnsson, M. Hammill, F. Short (eds.), *Simulation and Visualization on the Grid.*
14. E. Dick, K. Riemslagh, J. Vierendeels (eds.), *Multigrid Methods VI.*
15. A. Frommer, T. Lippert, B. Medeke, K. Schilling (eds.), *Numerical Challenges in Lattice Quantum Chromodynamics.*
16. J. Lang, *Adaptive Multilevel Solution of Nonlinear Parabolic PDE Systems.* Theory, Algorithm, and Applications.
17. B. I. Wohlmuth, *Discretization Methods and Iterative Solvers Based on Domain Decomposition.*
18. U. van Rienen, M. Günther, D. Hecht (eds.), *Scientific Computing in Electrical Engineering.*
19. I. Babuška, P. G. Ciarlet, T. Miyoshi (eds.), *Mathematical Modeling and Numerical Simulation in Continuum Mechanics.*
20. T. J. Barth, T. Chan, R. Haimes (eds.), *Multiscale and Multiresolution Methods.* Theory and Applications.
21. M. Breuer, F. Durst, C. Zenger (eds.), *High Performance Scientific and Engineering Computing.*
22. K. Urban, *Wavelets in Numerical Simulation.* Problem Adapted Construction and Applications.

23. L. F. Pavarino, A. Toselli (eds.), *Recent Developments in Domain Decomposition Methods.*

24. T. Schlick, H. H. Gan (eds.), *Computational Methods for Macromolecules: Challenges and Applications.*

25. T. J. Barth, H. Deconinck (eds.), *Error Estimation and Adaptive Discretization Methods in Computational Fluid Dynamics*.

26. M. Griebel, M. A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations*.

27. S. Müller, *Adaptive Multiscale Schemes for Conservation Laws*.

28. C. Carstensen, S. Funken, W. Hackbusch, R. H. W. Hoppe, P. Monk (eds.), *Computational Electromagnetics*.

29. M. A. Schweitzer, *A Parallel Multilevel Partition of Unity Method for Elliptic Partial Differential Equations*.

30. T. Biegler, O. Ghattas, M. Heinkenschloss, B. van Bloemen Waanders (eds.), *Large-Scale PDE-Constrained Optimization*.

31. M. Ainsworth, P. Davies, D. Duncan, P. Martin, B. Rynne (eds.), *Topics in Computational Wave Propagation*. Direct and Inverse Problems.

32. H. Emmerich, B. Nestler, M. Schreckenberg (eds.), *Interface and Transport Dynamics.* Computational Modelling.

33. H. P. Langtangen, A. Tveito (eds.), *Advanced Topics in Computational Partial Differential Equations.* Numerical Methods and Diffpack Programming.

34. V. John, *Large Eddy Simulation of Turbulent Incompressible Flows.* Analytical and Numerical Results for a Class of LES Models.

35. E. Bänsch (ed.), *Challenges in Scientific Computing - CISC 2002.*

36. B. N. Khoromskij, G. Wittum, *Numerical Solution of Elliptic Differential Equations by Reduction to the Interface.*

37. A. Iske, *Multiresolution Methods in Scattered Data Modelling.*

38. S.-I. Niculescu, K. Gu (eds.), *Advances in Time-Delay Systems.*

39. S. Attinger, P. Koumoutsakos (eds.), *Multiscale Modelling and Simulation.*

40. R. Kornhuber, R. Hoppe, J. Périaux, O. Pironneau, O. Wildlund, J. Xu (eds.), *Domain Decomposition Methods in Science and Engineering.*

41. T. Plewa, T. Linde, V.G. Weirs (eds.), *Adaptive Mesh Refinement – Theory and Applications.*

42. A. Schmidt, K.G. Siebert, *Design of Adaptive Finite Element Software.* The Finite Element Toolbox ALBERTA.

43. M. Griebel, M.A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations II.*

44. B. Engquist, P. Lötstedt, O. Runborg (eds.), *Multiscale Methods in Science and Engineering.*

45. P. Benner, V. Mehrmann, D.C. Sorensen (eds.), *Dimension Reduction of Large-Scale Systems.*

46. D. Kressner, *Numerical Methods for General and Structured Eigenvalue Problems.*

47. A. Boriçi, A. Frommer, B. Joó, A. Kennedy, B. Pendleton (eds.), *QCD and Numerical Analysis III.*

48. F. Graziani (ed.), *Computational Methods in Transport.*

49. B. Leimkuhler, C. Chipot, R. Elber, A. Laaksonen, A. Mark, T. Schlick, C. Schütte, R. Skeel (eds.), *New Algorithms for Macromolecular Simulation.*

50. M. Bücker, G. Corliss, P. Hovland, U. Naumann, B. Norris (eds.), *Automatic Differentiation: Applications, Theory, and Implementations.*

51. A.M. Bruaset, A. Tveito (eds.), *Numerical Solution of Partial Differential Equations on Parallel Computers.*

52. K.H. Hoffmann, A. Meyer (eds.), *Parallel Algorithms and Cluster Computing.*

53. H.-J. Bungartz, M. Schäfer (eds.), *Fluid-Structure Interaction.*

54. J. Behrens, *Adaptive Atmospheric Modeling.*

55. O. Widlund, D. Keyes (eds.), *Domain Decomposition Methods in Science and Engineering XVI.*

56. S. Kassinos, C. Langer, G. Iaccarino, P. Moin (eds.), *Complex Effects in Large Eddy Simulations.*

57. M. Griebel, M.A Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations III.*

58. A.N. Gorban, B. Kégl, D.C. Wunsch, A. Zinovyev (eds.), *Principal Manifolds for Data Visualization and Dimension Reduction.*

59. H. Ammari (ed.), *Modeling and Computations in Electromagnetics: A Volume Dedicated to Jean-Claude Nédélec.*

60. U. Langer, M. Discacciati, D. Keyes, O. Widlund, W. Zulehner (eds.), *Domain Decomposition Methods in Science and Engineering XVII.*

61. T. Mathew, *Domain Decomposition Methods for the Numerical Solution of Partial Differential Equations.*

62. F. Graziani (ed.), *Computational Methods in Transport: Verification and Validation.*

63. M. Bebendorf, *Hierarchical Matrices.* A Means to Efficiently Solve Elliptic Boundary Value Problems.

For further information on these books please have a look at our mathematics catalogue at the following URL: `www.springer.com/series/3527`

Monographs in Computational Science and Engineering

1. J. Sundnes, G.T. Lines, X. Cai, B.F. Nielsen, K.-A. Mardal, A. Tveito, *Computing the Electrical Activity in the Heart.*

For further information on this book, please have a look at our mathematics catalogue at the following URL: `www.springer.com/series/7417`

Texts in Computational Science and Engineering

1. H. P. Langtangen, *Computational Partial Differential Equations.* Numerical Methods and Diffpack Programming. 2nd Edition
2. A. Quarteroni, F. Saleri, *Scientific Computing with MATLAB and Octave.* 2nd Edition
3. H. P. Langtangen, *Python Scripting for Computational Science*. 3rd Edition
4. H. Gardner, G. Manduchi, *Design Patterns for e-Science*.
5. M. Griebel, S. Knapek, G. Zumbusch, *Numerical Simulation in Molecular Dynamics*.

For further information on these books please have a look at our mathematics catalogue at the following URL: `www.springer.com/series/5151`